P.1877/89

7 1989

informatyka

Felix Redmill
o niezawodności i bezpieczeństwie
Uruchamianie urządzeń mikroprogramowanych
Język symulacji na PC

WYDAWNICTWO NOT SIGMA

Cena 300 zł

informatyka

7 1989

Nr 7

Miesięcznik Rok XXIV
Lipiec 1989

Organ Komitetu
Naukowo-Technicznego NOT
ds. Informatyki

KOLEGIUM REDAKCYJNE:

Mgr Jarosław DEMINET,
dr inż. Wacław ISZKOWSKI,
mgr Teresa JABŁOŃSKA
(sekretarz redakcji),
Władysław KLEPACZ
(redaktor naczelny),
dr inż. Marek MACHURA,
dr inż. Wiktor RZECZKOWSKI,
mgr inż. Jan RYŻKO,
mgr Hanna WŁODARSKA,
dr inż. Janusz ZALEWSKI
(zastępca redaktora naczelnego).

PRZEWODNICZĄCY
RADY PROGRAMOWEJ:

Prof. dr hab.
Juliusz Lech KULIKOWSKI

Materiałów nie zamówionych redakcja
nie zwraca

Redakcja: 01-517 Warszawa, ul. Mickiewicza 18
m. 17, tel. 39-14-34

RSW „PRASA-KSIĄŻKA-RUCH"
PRASOWE ZAKŁADY GRAFICZNE
ul. Dworcowa 13, 85-950 BYDGOSZCZ
Zam. 1647/89. E-10
Obj. 4,0 ark. druk. Nakład 8950 egz.

ISSN 0542-9951. INDEKS 36124

Cena egzemplarza 300 zł
Prenumerata roczna 3600 zł

WYDAWNICTWO
NACZELNA ORGANIZACJA TECHNICZNA
CZASOPISM I KSIĄŻEK TECHNICZNYCH
SIGMA

00-950 Warszawa
skrytka pocztowa 1004
ul. Biała 4

W NUMERZE:

	Strona
Najnowsze zalecenia dotyczące bezpieczeństwa i niezawodności systemów komputerowych *Felix Redmill*	1
Rzetelność systemów komputerowych w zastosowaniach krytycznych *Janusz Zalewski*	4
Uruchamianie urządzeń mikroprogramowanych (1) *Marek Gondzio, Marek Pawłowski, Jacek Wytrębowicz*	6
Język symulacji systemów o działaniu ciągłym ze zdarzeniami dyskretnymi *Jacek Kuraś, Jacek Lembas, Marek Skomorowski*	9
Tomografia komputerowa *Marzena Tataj, Alina Toruń*	12
Przyspieszanie obliczeń przez ich rozpraszanie (3) Podsumowanie *Wacław Iszkowski*	15
Nowa generacja superkomputerów Cray (2). Cray-2 *Zdzisław Szczerbiński*	19
Z KRAJU Stanowisko Polskiego Towarzystwa Informatycznego w sprawie ochrony prawnej programów komputerowych	21
ZE ŚWIATA Wykonywanie programów DOS-a w systemie Unix Kto jest kim. Mike G. Rodd Architektura dla Ady Pogromca konkurentów – dBase IV	23
RECENZJE „Bazy danych – od koncepcji do realizacji"	29
TERMINOLOGIA Czym jest rzetelność systemu komputerowego?	30
NOWE KSIĄŻKI Wydawnictwa Naukowo-Techniczne	III okł.
W SKRÓCIE	IV okł.

Najbliższy numer będzie poświęcony w całości materiałom z konferencji PTI „40 lat informatyki w Polsce", która odbyła się w listopadzie 1988 roku w Warszawie. Znajdą się w nim wspomnienia konstruktorów pierwszych w Polsce maszyn liczących i pionierów oprogramowania – wybitnych naukowców:

- **Leona Łukaszewicza**
- **Romualda W. Marczyńskiego**
- **Antoniego Kilińskiego**
- **Antoniego Mazurkiewicza**
- **Jerzego Fietta**
- **Eugeniusza Bilskiego**
- **Marka J. Greniewskiego**
- **Krzysztofa Moszyńskiego**

WARUNKI PRENUMERATY

Prenumeratorzy zbiorowi – jednostki gospodarki uspołecznionej, instytucje i organizacje społeczne zamawiają prenumeratę dokonując wpłat wyłącznie na blankiecie „wpłata-zamówienie" (jest to „polecenie przelewu" rozszerzone dla potrzeb Wydawnictwa o część dotyczącą zamówienia). Blankiety te będą dostarczane dotychczasowym prenumeratorom przez Zakład Kolportażu. Nowi prenumeratorzy otrzymają je po zgłoszeniu zapotrzebowania (pisemnie lub telefonicznie) w Zakładzie Kolportażu.

Prenumeratorzy indywidualni – osoby fizyczne zamawiają prenumeratę dokonując wpłaty w UPT lub NBP na blankiecie NBP. Na odwrocie wszystkich odcinków blankietu należy wpisać tytuł czasopisma, okres prenumeraty, liczbę zamawianych egzemplarzy oraz wartość wpłaty. Wpłacać należy na konto: Państwowy Bank Kredytowy III/O Warszawa nr 370015-7490-139-11.

Prenumerata ulgowa – przysługuje wyłącznie osobom fizycznym – członkom SNT, studentom i uczniom szkół zawodowych. Warunkiem prenumeraty ulgowej jest poświadczenie blankietu wpłaty (przed jej dokonaniem) na wszystkich odcinkach pieczęcią Koła SNT, wyższej uczelni lub szkoły. Sposób zamawiania prenumeraty ulgowej jest taki sam jak prenumeraty indywidualnej. W prenumeracie ulgowej można zamówić tylko po jednym egzemplarzu każdego czasopisma. Uwaga! Miesięcznik „Aura" może być zamawiany w prenumeracie ulgowej również przez uczniów szkół ogólnokształcących.

Prenumeratę ze zleceniem wysyłki za granicę – zamawia się tak, jak prenumeratę indywidualną. Dodatkowo należy podać na blankiecie wpłaty nazwisko i dokładny adres odbiorcy. Cena prenumeraty ze zleceniem wysyłki za granicę jest dwukrotnie wyższa.

Wpłaty na prenumeratę przyjmowane są w terminach: do 10 listopada na każdy kwartał, I i II półrocze oraz cały rok następny; do 28 lutego na II, III i IV kwartał oraz II półrocze; do 31 maja na III i IV kwartał oraz II półrocze; do 31 sierpnia na IV kwartał. Zmiany w prenumeracie można zgłaszać pisemnie tylko w wyżej wymienionych terminach.

Informacji o prenumeracie udziela – Zakład Kolportażu Wydawnictwa NOT SIGMA (ul. Bartycka 20, 00-716 Warszawa), skr. poczt. 1004, 00-950 Warszawa, tel. 40-00-21, wewn. 248, 249, 293, 297, 299 lub 40-30-86 i 40-35-89.

Egzemplarze archiwalne czasopism – można nabyć za gotówkę w Klubie Prasy Technicznej, Warszawa ul. Mazowiecka 12 (tel. 26-80-16), lub zamówić pisemnie. Zamówienia na egzemplarze archiwalne czasopism przyjmuje: Zakład Kolportażu, Dział Handlowy, 00-950 Warszawa, skr. poczt. 1004 (tel. 40-37-31), na rachunek dla instytucji lub za zaliczeniem pocztowym dla osób fizycznych.

Cena prenumeraty

miesięczna		kwartalna		półroczna		roczna	
normalna	ulgowa	normalna	ulgowa	normalna	ulgowa	normalna	ulgowa
300 zł	60 zł	900 zł	180 zł	1800 zł	360 zł	3600 zł	720 zł

FELIX REDMILL
British Telecom International
Londyn, Wielka Brytania

P.1877/85

Najnowsze zalecenia dotyczące bezpieczeństwa i niezawodności systemów komputerowych

Powszechnie uważa się, że dziedzina wytwarzania oprogramowania przechodzi obecnie poważny kryzys. Zbyt często bowiem projekty programistyczne kończą się niepowodzeniem i są anulowane. Te, które są kontynuowane, często przedłużają się i przekraczają planowane koszty, a zdarza się, że nawet po zakończeniu nie spełniają wymagań użytkowników. Przyczyny tych niepowodzeń bywają różne, lecz jedną warto wymienić – jest to niestosowanie dyscypliny inżynierskiej, mimo iż wykonawcy oprogramowania lubią być nazywani inżynierami programowania (ang. *software engineers*).

Jest faktem, że inżynieria oprogramowania jest dziedziną młodą. Jak dotąd, nie określono jeszcze jasno ani jej standardów, ani nawet jej granic; nie ma też zawodowego kodeksu postępowania dla inżynierów programowania. W tych okolicznościach bardzo niewielu wykonawców oprogramowania stosuje się do wysokich standardów, a większość nie stosuje się w ogóle do żadnych.

Jedną z trudności, które muszą pokonać wykonawcy pragnący stosować w swej pracy profesjonalne standardy, są kłopoty ze znalezieniem udokumentowanych zaleceń mogących służyć za wzorce. Takich zaleceń, jednakże, nie brakuje. Jednym z ich źródeł jest Komitet Techniczny nr 7 (TC7) organizacji EWICS (ang. *European Workshop on Industrial Computer Systems*). Ten Komitet zajmuje się dziedziną wytwarzania oprogramowania od 1974 roku, poświęcając szczególną uwagę zagadnieniom bezpieczeństwa, niezawodności i zabezpieczeń (ang. *safety, reliability and security*). Pierwsze sześć zaleceń Komitetu opublikowano w formie książkowej w ubiegłym roku [5], a nowa książka zawierająca cztery najnowsze zalecenia jest w druku [6]. Te ostatnie zalecenia dotyczą miar oprogramowania oraz projektowania, oceniania i konserwowania systemów komputerowych z uwzględnieniem bezpieczeństwa.

W niniejszym artykule przedstawiono Komitet EWICS TC7 oraz omówiono krótko zalecenia zawarte w pierwszej książce i – bardziej szczegółowo – cztery najnowsze.

KOMITET TECHNICZNY NR 7 EWICS

Komitet Techniczny nr 7 EWICS zajmuje się od wielu lat niezawodnością i bezpieczeństwem systemów komputerowych [4], a od końca lat siedemdziesiątych czyni to ze wsparciem Komisji Wspólnot Europejskich CEC (ang. *Commission of the European Communities*). Statutowym zadaniem organizacji EWICS jest „*promowanie ekonomicznej i skutecznej realizacji programowanych systemów przemysłowych przez szkolenia, wymianę informacji oraz opracowywanie norm i zaleceń*". Zgodnie z tym stwierdzeniem, Komitet TC7 zajmuje się nie tylko opracowywaniem zaleceń, lecz także stanowi wartościowe forum wymiany informacji. Działając w wymienionych kierunkach, utrzymywano bardzo przyjazną i dynamiczną atmosferę spotkań, służącą rozwojowi świetnych stosunków między członkami Komitetu i wpływającą na wzrost produktywności.

Gdy w 1983 roku, ze względu na koniec okresu budżetowego, Komisja Wspólnot Europejskich zawiesiła pomoc dla różnych organizacji, wszystkie pozostałe komitety EWICS przestały działać wskutek braku funduszy. Jednakże dotychczas panujący entuzjazm i zapał członków Komitetu pozwoliły mu utrzymać się i kontynuować spotkania przez lata 1984–1985 bez wsparcia finansowego. W rzeczywistości właśnie w tym okresie zakończono prace nad zaleceniami przygotowywanymi wcześniej.

Obecnie Komitet odbywa spotkania tradycyjnie cztery razy w roku. Są one organizowane przez instytucje macierzyste członków w różnych krajach i gromadzą zwykle ok. 30 osób. Aktywni członkowie pochodzą z dwunastu krajów europejskich, włącznie z Polską, a korespondenci – z wielu innych krajów, jak USA, Japonia i Australia. Wszystkie spotkania są protokołowane, a praca odbywa się według planu określającego poszczególne zadania i okresowo weryfikowanego. Zadania są realizowane przez odrębne robocze grupy tematyczne, zgodnie z ustalonym harmonogramem.

W czasie swej działalności Komitet TC7 nawiązał współpracę z międzynarodowymi organizacjami normalizacyjnymi, które wykorzystują jego prace w opracowywanych przez siebie normach. Komitet TC7 sprawował także patronat nad szeregiem konferencji międzynarodowych [1], [7], będąc głównym organizatorem serii znanych i bardzo cenionych sympozjów SAFECOMP. Od 1979 roku odbyło się pięć takich sympozjów, a ich powodzenie przesądziło o przejściu na cykl spotkań odbywanych corocznie [2]. Następne sympozjum SAFECOMP odbędzie się w dniach 5–7 grudnia 1989 roku, w Wiedniu.

Rozpoczęte na początku lat siedemdziesiątych prace nad zaleceniami dla wykonawców systemów komputerowych zaowocowały opracowaniem sześciu dokumentów. Opublikowano je między październikiem 1981 a styczniem 1985 roku, jako oficjalne raporty Komitetu (ang. *position papers*). Ich zredagowana wersja stanowi treść książki [5] omówioną krótko w następnym punkcie.

W 1986 roku Komisja Wspólnot Europejskich przywróciła wsparcie dla Komitetu. Wiązało się to jednak z wykonaniem prac według uzgodnionego planu i z opracowaniem czterech nowych zaleceń w ciągu dwóch lat. Wskutek napiętego harmonogramu prac ucierpiała trochę działalność w sferze wymiany informacji, lecz zalecenia były gotowe na czas, tj. na koniec 1987 roku. Od tamtej pory ostatecznie je udoskonalo-

FELIKS REDMILL pracuje w firmie British Telecom International. Od dwudziestu lat zajmuje się równolegle wybranymi zagadnieniami telekomunikacji i informatyki, a obecnie kieruje wytwarzaniem systemów komputerowych do zastosowań telekomunikacyjnych. Jest autorem licznych artykułów i referatów publikowanych w czasopismach o zasięgu międzynarodowym i przedstawianych na specjalnych konferencjach. Opublikował też kilka książek na temat komputerów i central telefonicznych. Jest członkiem Komitetu Technicznego nr 7 EWICS (*European Workshop on Industrial Computer Systems*) i współautorem opracowanych przez ten Komitet zaleceń dotyczących wytwarzania systemów komputerowych do zastosowań krytycznych pod względem bezpieczeństwa i niezawodności. Z ramienia Komitetu redagował dwa zbiory tych zaleceń wydane w formie książkowej przez wydawnictwo Elsevier.

no, opublikowano w styczniu 1989 roku jako oficjalne raporty i przygotowano w zredagowanej postaci do wersji książkowej [6]. Ich treść omówiono w ostatnim punkcie artykułu.

SZEŚĆ PIERWSZYCH ZALECEŃ

Pierwszych sześć zaleceń, powstałych między październikiem 1981 a styczniem 1985 roku, dotyczy różnych aspektów okresu istnienia systemów komputerowych (ang. *system life cycle*). Część pierwsza każdego dokumentu stanowi wprowadzenie dla użytkowników i zawiera omówienie jego zakresu, przeznaczenia i zasadniczej tematyki, związków z pozostałymi zaleceniami oraz sposobów wykorzystania w różnych sytuacjach. Poniższy krótki opis, oparty na książce [5], przedstawia je w kolejności występującej w tej książce.

Pierwsze z nich, o nazwie „*A Guideline for the Documentation of Critical Computer Systems*" (zalecenia dotyczące dokumentowania systemów komputerowych w zastosowaniach krytycznych), obejmują zasady dokumentowania przez cały okres istnienia systemu, od fazy specyfikacji do eksploatacji i konserwacji. Podkreślono w nich, że opracowanie systemu docelowego jest tylko częścią całego przedsięwzięcia i przedstawiono wskazówki dotyczące dokumentowania zarządzania projektem oraz ogólnie – prowadzenia dokumentacji.

Kolejne zalecenia dotyczą tworzenia specyfikacji wymagań systemowych (ang. *system requirements specifications*). Wskazano w nich, że specyfikowanie wymagań należy do obowiązków nabywcy, choć w praktyce specyfikacje powstają często z pomocą wykonawców. Zalecenia zawierają obszerną listę tematów, które należy rozważyć w specyfikacji systemu komputerowego, wraz ze szczegółowymi wyjaśnieniami.

Trzeci dokument, o nazwie „*The Guideline for the Development of Critical Software*", obejmuje różne aspekty wytwarzania oprogramowania, jak: sprawy proceduralne, struktura oprogramowania, samokontrola programów, projektowanie szczegółowe i kodowanie oraz zagadnienia zależne od języka programowania. Te zalecenia zostały zawarte prawie w całości w międzynarodowej normie IEC 880 [3].

Następne zalecenia dotyczą sprzętu stosowanego w systemach komputerowych uwzględniających bezpieczeństwo. Zawierają one, przede wszystkim, analizę różnego rodzaju uszkodzeń występujących w komputerach. Na tej podstawie, w dalszej części, podano zasady projektowania – w kolejnych krokach – sprzętu samokontrolującego się i tolerującego uszkodzenia. W zakończeniu omówiono i porównano różne istniejące systemy i nowe rozwiązania w zakresie niezawodności sprzętu i tolerowania uszkodzeń. Treść zaleceń jest dość precyzyjna i w niektórych wypadkach zawiera dowody matematyczne.

Dwa ostatnie dokumenty dotyczą weryfikowania i atestowania (ang. *verification and validation*) oprogramowania w zastosowaniach krytycznych. Pierwszy z nich obejmuje zasady podstawowe i obowiązujące na każdym etapie wytwarzania oprogramowania. Drugi zawiera przegląd dostępnych technik weryfikowania i atestowania programów, według podziału na cztery kategorie: techniki analizy, techniki testowania, ocena niezawodności i testy baz danych. Każdą technikę omówiono podając jej cel, zwięzły opis, warunki stosowania, zalety i wady, skutki, związki z innymi technikami, ocenę, dostępne narzędzia i literaturę. Oba dokumenty są nieocenione dla osób zajmujących się weryfikowaniem i atestowaniem oprogramowania.

Wszystkie sześć zaleceń, zebrane w jednej książce, stanowią bardzo użyteczną pomoc dla wykonawców i dostawców systemów komputerowych, a szczególnie dla osób zaangażowanych w wytwarzanie oprogramowania.

OSTATNIE ZALECENIA

Na początku 1986 roku utworzono cztery grupy ekspertów do opracowania nowych zaleceń. Wersje robocze odpowiednich dokumentów dostarczono przed końcem 1987 roku Komisji Wspólnot Europejskich (CEC), która wspierała to przedsięwzięcie finansowo. W ciągu ubiegłego roku zalecenia udoskonalono i opublikowano jako oficjalne raporty (w styczniu br.). Obecnie są one przygotowywane do publikacji książkowej [6]. Poniżej omówiono w skrócie treść wszystkich czterech zaleceń.

Pomiary i zapewnienie jakości oprogramowania

W ostatnich latach, dzięki lepszym metodom zarządzania projektami, przestrzeganiu zasad inżynierskich, wprowadzeniu norm i użyciu technik zapewnienia jakości, znacznie ulepszono proces wytwarzania oprogramowania. Jednakże, trudności związane z pomiarami właściwości oprogramowania powodują, że jakość oprogramowania można określić tylko dla działającego (eksploatowanego) systemu, a nie można jej przewidzieć w procesie wytwarzania.

Opracowane zalecenia nie stanowią pełnego rozwiązania tego problemu, gdyż zagadnienie pomiaru właściwości oprogramowania jest jeszcze dalekie od rozwiązania, kładą natomiast podwaliny pod to rozwiązanie. Punktem wyjścia jest stwierdzenie, że nie samo oprogramowanie powinno być mierzone, lecz jego atrybuty. Dlatego istotne jest zidentyfikowanie najważniejszych atrybutów, na podstawie których można dokonywać oceny. W zaleceniach podzielono zbiór atrybutów na dwie kategorie – atrybuty jakościowe (ang. *quality attributes*) i atrybuty zasobnościowe (ang. *resource attributes*) – podając odpowiednie przykłady.

W celu właściwego sterowania projektem konieczne jest kontrolowanie atrybutów. W zaleceniach zaproponowano szereg reguł specyfikowania i kontroli atrybutów, a także podano przykłady najważniejszych atrybutów i pokazano, jak podzielić je na mierzalne podatrybuty do praktycznego zastosowania w projekcie. Ponadto, zaproponowano zasady ustalania kryteriów odbiorczych (ang. *acceptance criteria*). Bardzo obszerna bibliografia zawiera wykaz kilkudziesięciu międzynarodowych i krajowych norm, standardów i zaleceń praktycznych, dotyczących jakości oprogramowania.

Projektowanie systemów komputerowych z uwzględnieniem bezpieczeństwa

Choć konieczność przestrzegania zasad inżynierii oprogramowania wymusza na zespołach wykonawczych dużą sumienność w projektowaniu, projektowanie systemów z uwzględnieniem bezpieczeństwa wymaga jeszcze większej uwagi. Zagadnienie nie ogranicza się bowiem do funkcjonalności, niezawodności i konserwowalności, ale obejmuje również możliwe konsekwencje uszkodzeń i zapewnienie odporności na uszkodzenia. Aby umożliwić uwzględnienie w projekcie wszystkich czynników, proces projektowania podzielono w zaleceniach na cztery kroki, określając czynności wykonywane w każdym kroku.

Pierwszy krok polega na ogólnej analizie bezpieczeństwa i jest wstępem do faktycznego projektowania. W tym kroku należy wziąć pod uwagę nie tylko system komputerowy, lecz także sterowaną przezeń instalację oraz jej otoczenie. Dla systemu komputerowego należy ustalić cele bezpieczeństwa i zdefiniować kryteria ich spełnienia. Wymaga to przeprowadzenia pełnej analizy ryzyka (ang. *risk analysis*), wyróżnienia wszystkich możliwych stanów instalacji i zanalizowania przejść między stanami oraz przyczyn i skutków tych przejść.

Pierwszy krok można wykonać podczas przygotowywania specyfikacji funkcjonalnej. Krok drugi polega na analizie specyfikacji w celu sprawdzenia, czy wszystkie wymagania krytyczne dla bezpieczeństwa, określone w pierwszym kroku, są w pełni zdefiniowane i czy nie ma innych celów bezpieczeństwa, nie uwzględnionych w wymaganiach.

Krok trzeci i czwarty dotyczą samego projektowania. Zalecenia dotyczące trzeciego kroku obejmują tworzenie projektu i zarządzanie, osiąganie bezpieczeństwa przez niezawodność, ostrzeganie przed skutkami uszkodzeń, sprzężenie człowieka z komputerem (ang. *man-machine interface*) i aspekty komunikacyjne. Czwarty krok dotyczy metod atestacji projektu opracowanego w kroku trzecim.

Zbiór wskazówek dotyczących wszystkich czterech kroków stanowi tekst nieodzowny dla wszystkich projektantów systemów komputerowych z uwzględnieniem bezpieczeństwa.

Ocena bezpieczeństwa i niezawodności systemów komputerowych

Ten dokument dotyczy systemów, które są tak krytyczne, że wymagają wydania oceny i licencji przed wprowadzeniem do eksploatacji. Zawiera rady dla osób dokonujących oceny (ang. *assessors*), co do ich zadań, i dla wykonawców – czego mogą się spodziewać po tej ocenie. Dokument dzieli się na dwie części: właściwe zalecenia i kwestionariusz.

Zalecenia uzasadniają, że dokładna ocena powinna obejmować całe przedsięwzięcie, po pierwsze, aby możliwie najwcześniej wykryć zaistniałe problemy, a po drugie, ze względu na ograniczenia testowania i niedoskonałość metod wytwarzania oprogramowania. W celu umożliwienia dokładnej oceny, podzielono ją na cztery fazy: propozycja przedsięwzięcia, opracowanie, przyjęcie i zatwierdzenie, eksploatacja i konserwacja. Podczas oceny powstaje model będący podstawą przyjęcia (lub odrzucenia) systemu, zwany modelem bezpieczeństwa (ang. *safety case*).

W zaleceniach podkreśla się znaczenie zdefiniowania kryteriów odbioru ocenianego systemu. Te kryteria dotyczą całego okresu istnienia systemu i dotyczą stosowanych norm, kwalifikacji personelu, procedur zapewnienia jakości i wyników testów. Analizy kryteriów odbiorczych dokonuje się na dwóch poziomach – na poziomie środowiska i samego systemu komputerowego. W zaleceniach wyjaśniono rolę obu poziomów oraz omówiono czynniki istotne w ustalaniu kryteriów. Szczegółowo przedstawiono techniki oceny oraz model bezpieczeństwa. W zakończeniu, podano dokładne wskazówki dotyczące dokumentowania oceny.

Kwestionariusz dzieli się na pięć części odpowiadających poszczególnym fazom okresu istnienia systemu oraz trzy inne części dotyczące planowania i zarządzania, zatwierdzania, eksploatacji i konserwacji. Każda z nich dzieli się z kolei na rozdziały. Pytania w poszczególnych rozdziałach zawierają listy czynności dla osób oceniających i wykonawców, dotyczące odpowiednich części okresu istnienia projektu. Taki kwestionariusz jest bardzo dobrym narzędziem, ponieważ wykorzystując wskazówki z głównej części zaleceń osoba oceniająca może użyć go jako podstawy do planu oceny.

Konserwacja systemów komputerowych z uwzględnieniem bezpieczeństwa

W przeszłości, niewłaściwe specyfikowanie i złe metody projektowania i zarządzania przedsięwzięciami prowadziły do powstawania systemów, które wymagały poprawek, natychmiast po wprowadzeniu do eksploatacji, a później – znacznego wysiłku związanego z konserwacją. Jednocześnie, niewłaściwe zasady i procedury konserwacji powodowały pogorszenie jakości dokumentacji i zwiększenie złożoności systemu. Z tego względu, istotne jest stosowanie w konserwacji ścisłych zasad inżynierskich i procedur formalnych.

W zaleceniach podkreśla się, że każda proponowana zmiana w eksploatowanym systemie musi być najpierw poddana dokładnej analizie, a następnie zostać zaakceptowana na odpowiednim poziomie zarządzania. Jeżeli zmiana ma być zrealizowana, to należy ją traktować jako niezależne przedsięwzięcie i stosować sprawdzone metody projektowania i wytwarzania. Na zakończenie należy dokonać ponownej atestacji systemu i zaktualizować jego dokumentację.

Zalecenia oparto na użyciu tzw. diagramów pęcherzykowych (ang. *bubble diagrams*), które wyjaśniają istotę procesu konserwacji. Ilustrują cztery rodzaje konserwacji: zapobiegawczą, korekcyjną, doskonalącą i adaptacyjną (ang. *preventive, corrective, perfective, adaptive*). Każdy rodzaj konserwacji prowadzi jednak do zmiany w systemie i dlatego w stosowaniu wymaga tych samych zasad i kontroli.

Ponadto, zalecenia zawierają obszerny słownik wyjaśniający znaczenie wszystkich nazw użytych w diagramach oraz tzw. minispecyfikacje stanowiące opisy wszystkich czynności konserwacyjnych. Na opis czynności składają się wskazówki dotyczące sposobu zarządzania i szczegółowe procedury postępowania. Choć treść zaleceń dotyczy głównie systemów z uwzględnieniem bezpieczeństwa, co znalazło odbicie w stosowanej terminologii, podane reguły mogą być stosowane szerzej, a dokument można łatwo dostosować do potrzeb konserwacji innych rodzajów systemów w zastosowaniach krytycznych.

* * *

Dokonany powyżej przegląd czterech najnowszych zaleceń dotyczących odpowiednio pomiarów oprogramowania oraz projektowania, oceny i konserwacji systemów komputerowych z uwzględnieniem bezpieczeństwa może posłużyć jako wstęp do bardziej szczegółowej lektury odpowiednich dokumentów. Zainteresowani podstawami wytwarzania systemów komputerowych do zastosowań krytycznych mogą sięgnąć do zbioru pierwszych sześciu zaleceń. Wszystkie zalecenia opracował Komitet Techniczny nr 7 organizacji *European Workshop on Industrial Computer Systems*.

Tłumaczył i opracował
JANUSZ ZALEWSKI

LITERATURA

[1] Daniels B. K.: Proc. Safety and Reliability Society Symposium, Altrincham (UK), 10–12 November 1987. Elsevier, London, 1987
[2] Ehrenberger W.: Safety Related Computers in an Expanding Market. Proc. SAFECOMP'88. Pergamon Press, London, 1988
[3] International Electrotechnical Commission: Software for Computers in the Safety Systems of Nuclear Power Stations. IEC Publication 880, Geneva, 1986
[4] Rata J. M. A.: The Work of the Technical Committee on Reliability, Safety and Security of Industrial Computer Systems European Workshop on Industrial Computer Systems [1]
[5] Redmill F. J. (Ed): Dependability of Critical Computer Systems I. Elsevier, London, 1988
[6] Redmill F. J. (Ed.): Dependability of Critical Computer Systems II. Elsevier, London, 1989
[7] Zalewski J., Ehrenberger W.: Hardware and Software for Real Time Process Control. Proc. IFIP/IFAC/EWICS Working Conf., Warszawa, 30 May – 1 June 1988, North-Holland, Amsterdam, 1989.

Język symulacji systemów...

dokończenie ze s. 11

Planowana jest dalsza rozbudowa języka GODYS-PC. W kolejnej wersji przewiduje się wygodne w obsłudze menu i możliwość współpracy z myszą. Na życzenie użytkownika możliwe jest wygenerowanie wersji ze zmienionym zestawem funkcji standardowych.

LITERATURA

[1] Gordon G.: Symulacja systemów. WNT, Warszawa, 1974
[2] Jakubowski R., Król J.: Implementation of the simulation language based on functional graphs. Podstawy Sterowania, T. 2, 1972
[3] Jakubowski R., Król J.: Grafy funkcyjne w zastosowaniu do opisu i symulacji złożonych systemów dynamicznych. Podstawy Sterowania, T. 3, 1973
[4] Król J.: Symulacja cyfrowa złożonych systemów elektromechanicznych. Rozprawa doktorska. Politechnika Śląska, Gliwice, 1974
[5] Król J., Kuraś J., Lembas J.: Język symulacyjny dla komputerów ODRA 1300. Informatyka, nr 11, 1979
[6] Pofelski L., Skomorowski M.: Symulacja komputerowa. Uniwersytet Jagielloński. Kraków, 1983
[7] Król J., Kuraś J., Lembas J., Ślusarek M.: Implementacja języka do symulacji układów ciągłych ze zdarzeniami dyskretnymi. Podstawy Sterowania, T. 10, 1980
[8] Król J., Kuraś J., Lembas J., Rosek J.: Implementacja języka GODYS-6 dla komputera Riad-32. Raport roboczy. Instytut Informatyki, Uniwersytet Jagielloński, Kraków, 1985.

I Krajowa Konferencja Naukowa pn.

INŻYNIERIA WIEDZY
i
SYSTEMY EKSPERTOWE

odbędzie się we Wrocławiu w dniach 6–8 czerwca 1990 r.
Organizowana jest przez Instytut Sterowania i Techniki Systemów Politechniki Wrocławskiej i Komitet Automatyki PAN, przy współudziale Podkomitetu Automatyki PKPAiR NOT oraz Komisji Informatyki i Automatyki PAN Oddział Wrocław.

Tematyka konferencji:
- problemy i metody sztucznej inteligencji,
- metodologiczne podstawy inżynierii wiedzy,
- metody statystyczne w inżynierii wiedzy i systemach ekspertowych,
- metody i komputeryzacja badań eksperymentalnych,
- podstawy budowy systemów ekspertowych,
- ekspertowe systemy sterowania, identyfikacji i rozpoznania,
- zastosowania techniczne i biomedyczne.

Termin nadsyłania zgłoszeń oraz streszczeń referatów (o obj. 2 s. maszynopisu) upływa 30 listopada 1989 roku.

Adres organizatorów:
Dr Adam Grzech
Instytut Sterowania i Techniki Systemów Politechniki Wrocławskiej
ul. Janiszewskiego 11/17, 50–370 Wrocław

JANUSZ ZALEWSKI
Warszawa

Rzetelność systemów komputerowych w zastosowaniach krytycznych

Dzięki uprzejmości wydawnictwa *Elsevier*, możemy zrecenzować pierwszy zbiór zaleceń[1)] dotyczących wytwarzania i użycia systemów komputerowych w zastosowaniach krytycznych. Omawia je szerzej w wiodącym artykule bieżącego numeru *Felix Redmill*. Pewne wątpliwości polskich czytelników może wzbudzać tytuł tej publikacji: co znaczy angielski termin *dependability* i jak go przetłumaczyć na język polski?

Odsyłając zainteresowanych szczegółami do rozważań przeprowadzonych w dziale terminologii (str. 30), poinformuję tylko, że termin ten ukuli członkowie grupy roboczej 10.4 IFIP-u *(IFIP Working Group* 10.4 – *Reliable and Fault Tolerant Computing)*. W definicji podano, że jest to *właściwość systemu komputerowego, która umożliwia uzasadnione poleganie na usługach zapewnianych przez ten system*. Taka właściwość – można ją nazwać po polsku **pewnością, rzetelnością** lub jeszcze inaczej – przejawia się na zewnątrz przez **niezawodność** systemu komputerowego (ang. *reliability*) i jego bezpieczeństwo dla otoczenia (ang. safety). Jest oczywiste, że obie te cechy mają szczególne znaczenie w tzw. zastosowaniach krytycznych, a więc takich, które w niesprzyjających okolicznościach mogą prowadzić do znacznych strat ludzkich lub materialnych. Tego rodzaju zastosowań komputerów we współczesnym świecie jest bardzo wiele, np. w przemyśle jądrowym, energetycznym, chemicznym, metalurgicznym, w transporcie powietrznym, morskim, drogowym i kolejowym, w wojsku, telekomunikacji i innych dziedzinach życia. Choć istnieją różne opracowania dotyczące ogólnych zasad wytwarzania złożonych systemów komputerowych, to omawiana publikacja jest pierwszą tej rangi, odnoszącą się wyraźnie do zastosowań krytycznych – i to jest niewątpliwą zaletą.

Zalecenia zawarte w tej książce dzielą się bardzo logicznie na cztery zasadnicze rozdziały, zgodnie z podziałem cyklu wytwarzania systemów komputerowych na fazy: specyfikowania wymagań, wytwarzania oprogramowania, wytwarzania sprzętu oraz weryfikacji i atestacji systemu jako całości. **W rozdziale pierwszym**, nieco ogólniejszym, sformułowano wymagania dotyczące dokumentowania systemów komputerowych nie tylko podczas ich wytwarzania, lecz także w pełnym okresie istnienia, tzn. w fazach eksploatacji i konserwacji. Nie obejmuje on jednak tak ważnych przy wytwarzaniu faz, jak studium realizowalności i projektowanie, pomijając też inny ważny aspekt okresu istnienia systemu – zarządzanie konfiguracją. Mimo tych wad, zalecenia zawierają wiele istotnych wskazówek dotyczących dokumentowania systemów komputerowych w zastosowaniach krytycznych, przy założeniu, że taki system stanowi jedynie część większej instalacji. Dlatego też najważniejsze fragmenty tego rozdziału dotyczą zarządzania dokumentacją, opisu realizacji systemu z uwzględnieniem aspektów użytkowych, wielostronnej oceny systemu i zarządzania całym przedsięwzięciem. Poszczególne tematy, odpowiadające wybranym aspektom okresu istnienia systemu, dzielą się na podtematy (np. zarządzanie dokumentacją dotyczy jej struktury, standardów dokumentacyjnych i jej pielęgnowania), które obejmują najważniejsze elementy składowe, wyliczone w postaci punktów (dzielące się często na podpunkty) jako elementów tzw. list kontrolnych (ang. *checklist*). Przykładowo, jeden z punktów opisujących pielęgnowanie dokumentacji zaleca stosowanie do tego celu następujących narzędzi: edytory tekstów, generatory raportów, generatory spisu treści i indeksów, komparatory tekstów, systemy archiwowania, bazy danych i poczta elektroniczna.

Tak liczne sugestie co do uwzględniania wielu szczegółów dokumentowania mogą przyprawić o zawrót głowy, ale tylko pozornie. W rzeczywistości bowiem stanowią one jedynie dobrze uporządkowany układ odniesienia dla wykonawców systemu, którzy sami zadecydują, jakie z nich należy wziąć pod uwagę. Na tę dowolność właśnie kładzie się szczególny nacisk w zaleceniach, ponieważ poszczególne systemy komputerowe o najrozmaitszym przeznaczeniu są tak odmienne, że schematyczne dokumentowanie identycznych aspektów i w jednakowy sposób byłoby pozbawione sensu. Zalecenia, aczkolwiek bardzo szczegółowe, zostawiają jednak duży margines dowolności.

Jak w dokumentowaniu systemu komputerowego uwzględnić jego niezawodność i bezpieczeństwo – cechy trudne do wyodrębnienia i opisania na papierze? Autorzy zaleceń wybrnęli z tego problemu w sposób dość sprawny. Utrzymując ogólny schemat zaleceń, z podziałem na tematy, podtematy i punkty, w poszczególnych miejscach zaproponowano ujęcie tych elementów opisu, które są związane z niezawodnością i bezpieczeństwem systemu. Na przykład, w opisie systemu jako całości występują oddzielne listy kontrolne, złożone z punktów odnoszących się kolejno do zintegrowanego systemu (opis i ograniczenia zapewnienia bezpieczeństwa, obsługa defektów, zasady konserwacji i prowadzenia napraw, ostrzeganie przed niesprawnościami), samego sprzętu (wielopunktowy program zapewnienia niezawodności i odporności na defekty) i samego oprogramowania (osiąganie bezpieczeństwa i zabezpieczanie przed nieuprawnionym dostępem)

Reasumując trzeba stwierdzić, że niezwykle ważną cechą tego rozdziału zaleceń, choć odnosi się on do najmniej wdzięcznej strony działalności przy wytwarzaniu, eksploatowaniu i konserwowaniu systemu komputerowego, jest dostarczenie niemal pełnej informacji o możliwościach korzystania z dokumentacji. Po zapoznaniu się z tą partią materiału, wykonawca lub użytkownik powinien nabyć ogólnych umiejętności tworzenia dokumentacji w całym okresie istnienia systemu, przy utrzymaniu dużej dowolności w tworzeniu rzeczywiście potrzebnych dokumentów, uwzględniających elementy niezawodności i bezpieczeństwa.

Rozdział drugi obejmuje zalecenia dotyczące tworzenia specyfikacji wymagań systemowych. Przez specyfikację rozumie się w tej książce określenie wymagań, jakie powinien spełniać system komputerowy zdaniem nabywcy (ang. *the definition of what a customer wants a system to achieve*). Podkreśla się przy tym, co jest niesłychanie ważne, że specyfikacja wymagań systemowych powinna być jedynym środkiem porozumienia między nabywcą a dostawcą (lub wykonawcą) systemu. Ważne jest też, aby specyfikacja zawierała jedynie wymagania, a nie proponowała rozwiązań, gdyż mogą one wprowadzać istotne ograniczenia. Choć uważa się, że specyfikacja rzadko jest dokumentem ustalonym raz na zawsze (jest to oczywiste, ponieważ będąc przedmiotem zlecenia stanowi przedmiot ciągłych negocjacji między zleceniodawcą a zleceniobiorcą), powinna w pewnym momencie stać się dokumen-

[1)] F.J Redmill (Ed.). Dependability of Critical Computer Systems. Vol. 1. Elsevier Applied Science, London, 1988.

Dr inż. JANUSZ ZALEWSKI jest adiunktem w Instytucie Energii Atomowej w Świerku i zastępcą redaktora naczelnego „Informatyki". W pracach Komitetu Technicznego nr 7 EWICS bierze udział od 1980 roku.

tem zamrożonym. Od tej chwili wszelkie zmiany specyfikacji winny być przedmiotem formalnej procedury zmian (ang. *change control procedure*).

Tę część zaleceń oparto na założeniu, że pełna specyfikacja powinna odnosić się nie tylko do tzw. systemu docelowego (ang. *target system*), będącego przedmiotem zamówienia czy dostawy, lecz też do warunków produkcji i warunków eksploatacji tego systemu. Z tego powodu specyfikacja wymagań systemowych, oprócz wymagań dotyczących samego systemu docelowego, musi zawierać wymagania narzucane na środowisko eksploatacyjne tego systemu, a także na środowisko produkcyjne, tzn. całe przedsięwzięcie (ang. *project*). Zasadniczą część tego rozdziału stanowią więc zalecenia odpowiadające wymienionemu założeniu, określające, jak formułować wymagania dla systemu docelowego, jego środowiska oraz wymagania odnoszące się do całego cyklu produkcyjnego (przedsięwzięcia) i środowiska, w jakim jest realizowany.

W dziale dotyczącym systemu docelowego najciekawsze są zalecenia związane z wymaganiami decydującymi o rzetelności systemu, a więc uwzględniające takie cechy, jak niezawodność, bezpieczeństwo i poufność, lecz także – adaptowalność, dyspozycyjność i konserwowalność systemu. W pozostałych trzech działach zasadniczych zalecenia dotyczą tylko cech specyficznych, np. dla środowiska eksploatacyjnego szczególnie istotne jest bezpieczeństwo, a dla całego przedsięwzięcia (tzn. cyklu produkcyjnego) i jego środowiska – zagadnienie kontroli jakości. Oczywiście, główną część specyfikacji powinny stanowić tradycyjne wymagania dotyczące funkcji, wydajności i sprzężeń systemu docelowego – co też wyraźnie podkreślono w zaleceniach. Warto też zwrócić uwagę, że podobnie jak w zaleceniach dokumentacyjnych, ogólne ujęcie zaleceń na specyfikację wymagań systemowych może posłużyć do określenia bardziej szczególnych zaleceń na poziomie określonej instytucji lub nawet pojedynczego kontraktu (w wypadku, gdy istotne będą tylko niektóre z nich).

Kolejne dwa rozdziały zawierają zalecenia dotyczące wytwarzania oprogramowania i sprzętu do zastosowań krytycznych.

Zbiór zaleceń na wytwarzanie oprogramowania był już znany wcześniej, ponieważ został zaadaptowany przez Międzynarodową Komisję Elektrotechniczną, do stosowania w energetyce jądrowej i opublikowany jako norma IEC 880, pn. *Software for Computers in the Safety Systems of Nuclear Power Stations* (Genewa, 1986). Zawiera on wskazówki dotyczące zarówno projektowania, jak i implementacji oprogramowania, dzieląc cały proces wytwarzania na kilka bardziej szczegółowych etapów; pomija natomiast zagadnienie testowania, weryfikacji i licencjonowania. Główne założenie, jakie przyświecało opracowaniu tych zaleceń sprowadza się do żądania, aby zminimalizować w wytwarzanym oprogramowaniu występowanie defektów i uczynić je jak najbardziej czytelnym. Dzięki temu, oczywiście, oprogramowanie będzie bardziej niezawodne i bezpieczniejsze.

Budowa tego dokumentu jest nieco inna niż pozostałych. Zalecenia umieszczono w ponad dwudziestu tabelach, grupując je według podziału procesu produkcyjnego na fazy i szeregując w kolejności od najogólniejszych do najbardziej szczegółowych. Każdemu zaleceniu przypisano kilka dodatkowych informacji, jak stopień ważności (priorytet), wskazówki na temat sposobu implementacji (system operacyjny, program usługowy, program użytkowy, dokumentacja programu, sprzęt), sprawdzalność wprowadzenia (w dokumentacji, w kodzie, podczas testowania, trudno sprawdzalne) i komentarz nt. celowości stosowania zalecenia. Wprost do zagadnień rzetelności oprogramowania odnosi się tylko jedna grupa zaleceń, mianowicie traktująca o samokontroli programów; uwzględniono w niej tzw. kontrole wiarygodności (ang. *plausibility checks*), meldunki o błędach, ochronę pamięci i przeciwdziałanie defektom.

Rozdział dotyczący wytwarzania sprzętu do zastosowań krytycznych ma w przeciwieństwie do rozdziałów poprzednich formę wykładu. Wyróżniono w nim dwie metody osiągania dużej rzetelności, odpowiadające jej dwom istotnym aspektom: przez maksymalne zwiększenie prawdopodobieństwa, że system komputerowy nie ulegnie uszkodzeniu (aspekt niezawodności) i przez zapewnienie, że w wypadku awarii system przejdzie do stanu bezpiecznego (aspekt bezpieczeństwa). Po przedstawieniu podziału uszkodzeń ze względu na przyczyny i skutki (statyczne i przejściowe, systematyczne i losowe, pojedyncze i wielokrotne, niebezpieczne i niegroźne, ujawnione i nieujawnione), wyodrębniono grupę uszkodzeń prowadzącą do zagrożenia bezpieczeństwa i zaproponowano zasady przeciwdziałania poszczególnym uszkodzeniom z tej grupy, oparte na szczegółowej analizie ich rodzajów.

Dwa ostatnie rozdziały dotyczą, odpowiednio, metodyki i technik weryfikowania i atestowania systemów komputerowych w zastosowaniach krytycznych, przede wszystkim pod względem rzetelności oprogramowania. Główną i niezbywalną zasadą weryfikacji i atestacji jest całkowita niezależność i rozłączność zespołów: wykonawczego oraz weryfikacyjno-atestacyjnego. Weryfikacja – w rozumieniu tego dokumentu – polega na porównaniu zgodności wyniku każdego kolejnego etapu wytwarzania systemu komputerowego z ustaleniami dokonanymi na poprzednim etapie. Atestacja (ang. *validation*) natomiast stanowi stwierdzenie zgodności całego systemu przekazywanego do eksploatacji z wymaganiami określonymi w jego specyfikacji. Zalecenia podane w tych rozdziałach stanowią szereg wskazówek co do preferowanych technik w kolejnych fazach wytwarzania wyrobu, jak specyfikowanie, projektowanie, kodowanie, testowanie modułów, integrowanie modułów, integrowanie sprzętu i oprogramowania i atestowanie całego systemu.

Książka stanowiąca zbiór zaleceń dotyczących wytwarzania systemów komputerowych do zastosowań krytycznych jest niewątpliwie ważnym i potrzebnym przewodnikiem dla projektantów, wykonawców i nabywców takich systemów. Problem polega na tym, czy spełnia ich oczekiwania. Trzeba więc powiedzieć, że nie jest ona gotowym zbiorem recept i pod tym względem niejednego czytelnika może rozczarować. Jest natomiast znakomitym przeglądem problematyki rzetelności systemów komputerowych w zastosowaniach krytycznych i jako taka stanowi świetny punkt odniesienia, bądź do opracowania własnego zbioru zaleceń dla konkretnego zastosowania, bądź do wyselekcjonowania tych cech budowanego systemu komputerowego, które będą w danym zastosowaniu istotne.

Oprócz tej istotnej zalety, książka ma wiele wad. Za największą uznałbym dużą niejednorodność tekstu. Mimo pozornie identycznej budowy logicznej wszystkich rozdziałów, zasadniczą część każdego rozdziału, tzn. same zalecenia, przedstawiono bardzo różnie – albo w formie wykazów, albo opisów werbalnych, albo w formie wykładu, albo niesystematycznej i miejscami chaotycznej opowieści. Bardzo zróżnicowany jest również stopień szczegółowości poszczególnych zaleceń. Wyraźnie brakuje odrębnego rozdziału dotyczącego integrowania sprzętu z oprogramowaniem do zastosowań krytycznych. Jest to etap najbardziej czuły, wymagający szczególnej uwagi od wykonawców i ścisłego przestrzegania przez nich bardzo surowego reżimu produkcyjnego. Innym zarzutem, który można postawić tej książce, jest jej częściowa nieaktualność – najstarsze zalecenia pochodzą z początku lat osiemdziesiątych. Od tego czasu technologia wytwarzania systemów komputerowych znacznie się zmieniła, powstało wiele nowych metod, które z oczywistych przyczyn nie mogły być uwzględnione w tej publikacji. Z drugiej strony jednak, taka sytuacja zaświadcza o wyjątkowej zasadności i trwałości tych zaleceń – żadne normy, a tym bardziej w tak ważnej dziedzinie zastosowań jak systemy krytyczne, nie mogą powstawać z dnia dzień, lecz muszą być oparte na solidnych, dobrze sprawdzonych podstawach. W sumie książka ta zawiera dużo ciekawego materiału, który będzie z całą pewnością przydatny w wielu zastosowaniach praktycznych.

Ekspansja japońska

Po podsumowaniu zeszłorocznych wyników światowej produkcji układów scalonych, której łączna wartość wyniosła 50,2 mld dolarów, okazało się, że udział Japonii wyniósł aż 50%. W porównaniu z 1987 r. oznacza to wprawdzie stosunkowo niewielki, bo tylko 2% wzrost, ale świadczy o dalszym umacnianiu się dominacji japońskiej w tej kluczowej dziedzinie technologii.

Przy okazji warto przypomnieć, że Japonia już w 1985 r. wyparła USA z pierwszego miejsca na liście krajów produkujących układy scalone. O sile ekspansji japońskiej jeszcze dobitniej świadczy fakt, że na rynku USA – w pozycji pamięci o pojemności 1 Mb, które obecnie są probierzem zaawansowanej technologii – udział dostaw japońskich wzrósł w ub. r. o dalsze 5%, pokrywając w tej pozycji 20% zapotrzebowania. Świadczy to, że protekcjonistyczne działania rządu USA są wobec Japonii w dalszym ciągu mało skuteczne.

MAREK GONDZIO
MAREK PAWŁOWSKI
JACEK WYTRĘBOWICZ
Instytut Informatyki
Politechnika Warszawska

Uruchamianie urządzeń mikroprogramowanych (1)

Dwa lata temu w Informatyce [1] omówiono problemy projektowania urządzeń mikroprogramowanych. Po zmontowaniu nowo zaprojektowanego urządzenia konieczne jest jego uruchomienie, tzn. sprawdzenie poprawności jego działania. Uruchamianie urządzeń mikroprogramowanych można porównać do uruchamiania systemów mikroprocesorowych; polega ono na:

- sprawdzeniu prawidłowości montażu poszczególnych bloków urządzenia,
- weryfikacji poprawności jego oprogramowania.

Aby ułatwić i przyspieszyć uruchamianie systemów mikroprocesorowych, opracowano dużą grupę narzędzi zarówno sprzętowych, jak i programowych. Do narzędzi sprzętowych należą emulatory układowe dołączane do systemów MDS, a do programowych – edytory, translatory i programy uruchomieniowe. Narzędzia te są tak opracowane, że w większości wypadków mogą być stosowane w dowolnym systemie mikroprocesorowym (z dokładnością do użytego mikroprocesora).

Właściwości tej nie mają układy mikroprogramowane. Urządzenia mikroprogramowane są zwykle projektowane do realizacji specyficznych zadań, dlatego różnią się zarówno architekturą wewnętrzną, jak i długością słowa sterującego ich działaniem [1]. Konieczne są więc specjalne narzędzia do uruchamiania części sprzętowej i jej oprogramowania. Muszą one zapewnić m. in. możliwość:

- bezbłędnego wygenerowania zawartości pamięci mikroprogramu (o organizacji kilkuset słów kilkudziesięciobitowych),
- modyfikowania zawartości pamięci mikroprogramu podczas uruchamiania urządzenia,
- śledzenia i zatrzymywania działania urządzenia mikroprogramowanego w zadanym mikrorozkazie.

PROBLEMY URUCHAMIANIA URZĄDZEŃ MIKROPROGRAMOWANYCH

Długie słowo mikrorozkazowe [1] oraz jego zmienna budowa dla różnych urządzeń jest przyczyną kłopotów związanych z określeniem zawartości pamięci mikroprogramu. Można ich uniknąć wykorzystując do tego celu metaasembler (np. MIC) umożliwiający symboliczne zapisanie zawartości pamięci mikroprogramu. Jednak nawet najbardziej staranne jej przygotowanie nie daje konstruktorowi pewności, że oprogramowanie urządzenia nie zawiera błędu. Konieczny jest więc etap doświadczalnej weryfikacji mikroprogramu.

Ze względu na szybkość działania układów mikroprogramowanych, sterowanie nimi jest zapisywane w bipolarnych pamięciach stałych PROM. Zmiana mikroprogramu wymaga więc zwykle przeprogramowania kilku lub kilkunastu pamięci PROM (w zależności od liczby zmienianych bitów sterowania). Ponieważ każda z komórek pamięci PROM może być programowana tylko raz i przywrócenie jej poprzedniego stanu jest niemożliwe, efektem częstych zmian w mikroprogramie może być stos zniszczonych układów pamięci stałych.

Pamięci PROM można zastąpić reprogramowalnymi pamięciami EPROM o tym samym znaczeniu końcówek. Niestety, są one kilkakrotnie wolniejsze niż pamięci PROM, dlatego przy takim postępowaniu można uruchomić urządzenie mikroprogramowane tylko w sposób funkcjonalny, tzn. bez weryfikacji parametrów czasowych. Niemożliwe jest sprawdzenie wszystkich funkcji urządzenia, gdy są ściśle określone zależności czasowe dla współpracy urządzenia z otoczeniem. Co prawda pojawiły się już układy EPROM o czasie dostępu takim samym jak pamięci PROM (np. CY7C261), ale są one dość drogie i na naszym rynku niedostępne.

Nawet, jeżeli jest możliwe zastąpienie pamięci PROM pamięcią EPROM, to wymazywanie, wprowadzenie poprawek i programowanie pamięci EPROM może pochłaniać czas rzędu kilkudziesięciu minut. Aby przyspieszyć ten proces, można podczas uruchamiania urządzenia użyć pamięci RAM zamiast PROM. Wymaga to jednak sporej rozbudowy urządzenia o układy umożliwiające wprowadzenie danych do pamięci sterującej o słowie kilkudziesięciobitowym, z dodatkowego zewnętrznego urządzenia dostarczającego informację w postaci 8-bitowej. Te dodatkowe układy, konieczne tylko podczas uruchamiania urządzenia, muszą być przewidziane w procesie jego projektowania. Zwiększa to koszt danego urządzenia.

Można sobie jednak wyobrazić, że pamięć RAM symulująca pamięć mikroprogramu może być umieszczona poza uruchamianym urządzeniem i będzie używana do uruchamiania wielu podobnych urządzeń. W ten sposób powstały symulatory pamięci stałych, np. moduł SYMROM w systemie MSWP.

Żeby symulator PROM był uniwersalny, musi mieć następujące cechy:

- bardzo długie słowo mikrorozkazowe – 128–256 bitów;
- krótki czas dostępu – 25–100 ns, odpowiadający parametrom najczęściej używanych pamięci PROM;
- możliwość dołączenia wewnętrznego rejestru na wyjściu symulowanej pamięci (wynika to z faktu istnienia PROM z rejestrami w jednej obudowie);
- łatwość dołączania symulatora do uruchamianego urządzenia, uzyskana przez zastosowanie specjalnych układów sond wkładanych w podstawki PROM.

Moduł SYMROM umożliwia symulowanie pamięci stałej o pojemności 512 × 64 bitów o czasie dostępu 150 ns. Jest on częścią systemu MSWP i jego działaniem steruje program umożliwiający m. in. załadowanie i edycję pamięci symulowanej.

Najczęściej jednak symulator pamięci PROM jest częścią większego systemu specjalnie zaprojektowanego do wspomagania uruchamiania urządzeń mikroprogramowanych. Przykładami takich systemów mogą być SYSTEM 29 firmy Advanced Micro Devices i STEP2 firmy STEP ENGINEERING. Nie są to najnowsze systemy między innymi z powodu oszczędności elementów i mocy zastosowano w nich możliwość rekonfiguracji pamięci, tzn. zmiany jej organizacji polegającej na skróceniu słowa na rzecz zwiększenia liczby słów. Rozwiązanie to dodatkowo zwiększa czas dostępu dla symulowanej pamięci i nie jest stosowane w nowszych wersjach systemów uruchomieniowych, np. STEP27 i DS3700 [2].

SYSTEM 29 umożliwia symulowanie pamięci PROM o długości słowa do 128 bitów. Organizacja pamięci może być zmieniana z 2048 słów 128-bitowych na 4096 słów 64-bitowych. Czas dostępu do pamięci w starszych wersjach wynosi 250 ns, a w nowszych – kilkadziesiąt nanosekund.

W systemie STEP2 pamięć symulatora jest podzielona na trzy moduły, które mogą tworzyć dwie niezależne tablice (tzn. można z jej użyciem uruchamiać dwa urządzenia równocześnie). Każdy z modułów można skonfigurować na 4K × 8 bitów, 2K × 16 bitów lub 1K × 32 bity. Oznacza to, że słowo mikrorozkazowe może mieć maksymalną długość 96 bitów.

Sama symulacja pamięci PROM umożliwia szybką modyfikację pamięci sterującej, ale nie wystarcza do równie szybkiej lokalizacji

błędów w jej zawartości. Konieczne są więc mechanizmy umożliwiające śledzenie kolejności mikrorozkazów odczytywanych z pamięci mikroprogramu, z możliwością zatrzymania urządzenia podczas odczytywania wybranego słowa (tzn. pułapka adresowa) lub wystąpienie określonego stanu logicznego w wybranym punkcie uruchamianego układu.

Podczas działania mikroprogramu może zdarzyć się, że jest wykonywana nieprawidłowa sekwencja mikrorozkazów, będąca wynikiem błędnego działania układu sterującego [1]. Do zlokalizowania przyczyny jej wystąpienia jest konieczny ślad wykonania mikroprogramu. Może on być zapisany w pamięci śladu, zawierającej adresy n – ostatnio wykonanych mikrorozkazów. Wyznaczenie mikrorozkazu, który zakłócił prawidłową sekwencję słów sterujących urządzeniem mikroprogramowanym polega na porównaniu prawidłowej sekwencji z sekwencją zapisaną w pamięci śladu. Po ponownym zainicjowaniu działania urządzenia mikroprogramowanego, z tymi samymi co poprzednio danymi wejściowymi, można ustawić pułapkę na wyznaczony mikrorozkaz. Zatrzymanie działania urządzenia w sytuacji prowadzącej do błędu pozwala na statyczną lokalizację uszkodzenia urządzenia. Lokalizacja ta może być przeprowadzana np. za pomocą sondy logicznej.

Duża pojemność pamięci śladu oraz możliwość ustawienia wielu pułapek znacznie przyspiesza proces uruchamiania urządzenia mikroprogramowanego. SYSTEM 29 jest wyposażony w pamięć śladu o pojemności 256 słów 12-bitowych, w której jest zapisywany ślad przejścia mikroprogramu przez ostatnie 256 punktów testujących wyznaczonych przez użytkownika. System ten zawiera tylko jeden rejestr pułapki, tzn. dopuszcza określenie tylko jednego adresu mikrorozkazu, którego wybranie przez układ sterujący powinno zatrzymać działanie urządzenia. System STEP2 jest wyposażony w trzy rejestry pułapek, po jednym na każdy moduł pamięci symulującej PROM, oraz w licznik cykli lub zdarzeń pozwalający na zatrzymanie zegara uruchamianego urządzenia po n-krotnym przejściu przez wskazany adres.

W pierwszych wersjach pamięć śladu zawierała tylko adresy mikrorozkazów wywoływanych przez uruchamiany układ sterujący. W najnowszych rozwiązaniach zbiera się również ślad z kilku dodatkowych sond dwu- lub trójwartościowych, umieszczanych przez użytkownika w dowolnych punktach uruchamianego urządzenia. Stan tych sond może być wykorzystany do określenia warunku zatrzymania działania urządzenia mikroprogramowanego.

Zatrzymanie pracy uruchamianego urządzenia jest możliwe przez zablokowanie sygnału zegara. Konieczna jest więc budowa odpowiedniego układu zegara w urządzeniu uruchamianym. Podczas weryfikowania parametrów dynamicznych zaprojektowanego urządzenia często trzeba modyfikować czas trwania wysokiego lub niskiego poziomu sygnału zegara. Aby taka modyfikacja była prosta i szybka, należałoby ją zrealizować w sposób programowy. Jest to możliwe, jeśli system wspomagający uruchamiania urządzeń mikroprogramowanych zawiera blok programowanego zegara.

W systemach SYSTEM 29 i STEP2 oraz w nowszych wersjach, tj. STEP27 i DS3700 nie ma możliwości programowej modyfikacji parametrów zegara.

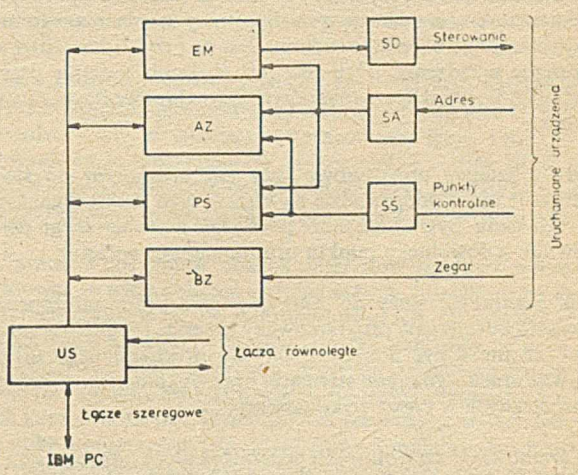
Wszystkie wymienione problemy uruchamiania mikroprogramowanych urządzeń cyfrowych zostały rozwiązane w zaprojektowanym, wykonanym i eksploatowanym w Instytucie Informatyki Politechniki Warszawskiej systemie MIDES, wspomagającym uruchamianie urządzeń mikroprogramowanych.

STRUKTURA SYSTEMU MIDES

Na system MIDES składają się trzy elementy:
– system mikroprocesorowy SUMUS2, realizujący omówione wyżej zadania,
– system IBM PC XT/AT jako system nadrzędny przekazujący zadania do wykonania systemowi SUMUS2,
– specjalistyczne oprogramowanie wspomagające uruchamianie układów mikroprogramowanych.

Realizowane są dwie wersje systemu MIDES, o zbliżonych właściwościach funkcjonalnych, a różniących się elementami stosowanymi do ich budowy. MIDES-P to wersja profesjonalna, przeznaczona do pracy w czasie rzeczywistym; do jej budowy stosuje się układy scalone produkowane w krajach zachodnich. MIDES-T to wersja dydaktyczna, przeznaczona do uruchamiania struktur mikroprogramowanych o krótszym słowie mikrorozkazowym i zmniejszonej szybkości pracy podczas uruchamiania. Wersja ta jest zbudowana głównie z układów scalonych produkowanych w krajach RWPG.

Konstruktor prowadzi dialog z systemem MIDES za pośrednictwem monitora i klawiatury mikrokomputera IBM PC. Bezpośrednią kontrolę nad uruchamianym urządzeniem przejmuje natomiast część sprzętowa systemu. Jej zasadniczymi blokami funkcjonalnymi są:
– emulator pamięci stałej *EM*,
– analizator zdarzeń *AZ*,
– pamięć śladu *PŚ*,
– blok zegara *BZ*,
– układ sterujący *US*.
Strukturę połączeń międzyblokowych pokazano na rys. 1.



Rys. 1. Schemat blokowy systemu SUMUS

Blok *EM* umożliwia emulację pamięci stałej zawierającej treść mikroprogramu. Po zatrzymaniu pracy uruchamianego urządzenia, konstruktor może odczytywać i modyfikować zawartość pamięci mikroprogramu. Operacje te nie wpływają na stan wyjściowego słowa danych, który jest niezmienny i równy ostatnio czytanej wartości, podczas wykonywania mikroprogramu. Blok *EM* łączy się z uruchamianym urządzeniem przez sondy: adresową *SA* i danych *SD*. Sondy są elementami aktywnymi, ich celem jest wzmocnienie odbieranych sygnałów i dopasowanie falowe linii łączących system z uruchamianym urządzeniem.

Blok *EM* jest zrealizowany z modułów o pojemności 2K słów (lub 1K słów, w zależności od wykonania) 64-bitowych. Każdy moduł zwiększa długość słowa mikrorozkazowego o kolejne 64 bity. W urządzeniu można zainstalować do czterech takich modułów, co daje maksymalną pojemność 2K słów 256-bitowych. Każdy kolejny bajt danych słowa wyjściowego może być wprowadzany w stan dużej impedancji odrębnym sygnałem sterowanym z uruchamianego urządzenia.

Możliwe jest emulowanie pamięci z rejestrem na wyjściu. W tym celu wystarczy wydać odpowiednie polecenie systemowi MIDES i automatycznie blok *EM* zostanie wyposażony w taki rejestr. Ponadto konstruktor może programowo (odpowiednim poleceniem) zerować niewykorzystywane bardziej znaczące bity adresowe pamięci *EM*.

Blok analizatora zdarzeń *AZ* służy do wykrywania zdefiniowanych zdarzeń lub sytuacji wyjątkowej. Sytuacją wyjątkową jest wyjście poza przestrzeń adresową mikroprogramu. Blok *AZ* zawiera mechanizmy umożliwiające rozpoczęcie wykonywania mikroprogramu od zadanego mikrorozkazu i zatrzymanie jego pracy po wystąpieniu jednego z poniższych zdarzeń:
- wykonanie N mikrorozkazów, gdzie $N \in \langle 1,256 \rangle$,
- wejście na pułapkę zwykłą (można zdefiniować dowolną liczbę pułapek, przypisując je dowolnym mikrorozkazom),
- M-krotne przejście przez pułapkę z licznikiem, gdzie $M \in \langle 1,256 \rangle$,
- odczytanie słowa pamięci, które nie zawiera mikrorozkazu,
- osiągnięcie zgodności ze wzorcem na wybranych liniach śladu,
- pojawienie się sygnału zatrzymania pochodzącego z uruchamianego urządzenia,
- wydanie polecenia zatrzymania przez operatora.

Blok pamięci śladu *PŚ* umożliwia bieżące rejestrowanie informacji podczas wykonywania mikroprogramu. Zbieranie danych nie ogranicza szybkości pracy uruchamianego urządzenia. Po zatrzymaniu pracy, konstruktor może przeanalizować zapamiętane adresy wykonywanych mikrorozkazów i stan dowolnie wybranych punktów w uruchamianym sprzęcie. Informacje są podawane do bloku *PŚ* z sondy adresów *SA* (11 bitów) i sond śladu *SŚ* (16 bitów).

Zrealizowano dwa rodzaje sond śladu: dwu- i trójwartościowe. Sonda dwuwartościowa rozróżnia wysoki i niski poziom TTL. Sonda trójwartościowa rozpoznaje ponadto niezdefiniowany poziom TTL (wiszące wejście, stan dużej impedancji, napięcie z zakresu ⟨0,9V; 2,0V⟩).

Blok PŚ umożliwia zarejestrowanie śladu wykonania 1024 mikrorozkazów. Efektywność procesu uruchamiania zwiększa funkcja selektywnego zbierania śladu. Przypisanie znacznika śledzenia wybranym mikrorozkazom ułatwia późniejszą analizę pracy uruchamianego urządzenia wykonującego określone zadania. Jest na przykład możliwa analiza grafu pracy urządzenia, jeżeli znacznik śledzenia zostanie przypisany wszystkim mikrorozkazom rozpoczynającym liniowe fragmenty mikroprogramu.

Oprócz zadanej programowo listy mikrorozkazów do śledzenia, konstruktor może doprowadzić na wejście tego bloku sygnał żądania rejestracji śladu. Sygnał ten może pochodzić z zewnętrznego detektora błędów lub z dowolnego punktu uruchamianego sprzętu.

Blok zegara *BZ* służy do zatrzymywania pracy mikroprogramu. Zawiera on wewnętrzny programowany generator sygnału zegarowego. Sygnał ten może być doprowadzony do uruchamianego urządzenia i sterować pracą jego części sterującej oraz wykonaniem mikrooperacji synchronicznych w części wykonawczej..

Programowanie częstotliwości generatora może odbywać się z kwantem 50 ns, a okres sygnału zegarowego może zmieniać się w zakresie od 100 ns do 25,6 μs. Zastosowanie zewnętrznego strojonego oscylatora pozwala uzyskać sygnał zegarowy o dowolnej częstotliwości i wypełnieniu.

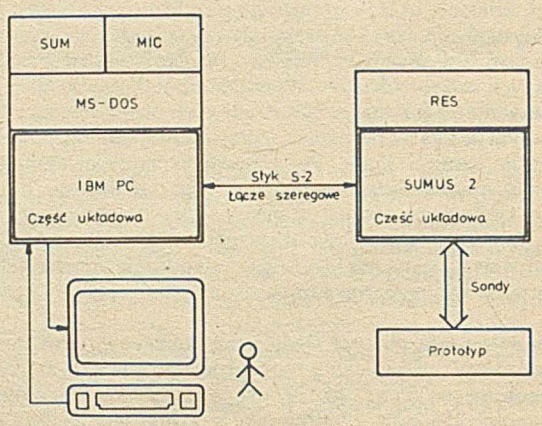
Oczywiście uruchamiane urządzenie mikroprogramowane może korzystać z własnego układu zegara przy uruchamianiu. Wskazane jest wówczas, aby układ ten wstrzymywał swoją pracę pod wpływem sygnału żądania zatrzymania, pochodzącego z bloku *BZ*.

Blok sterujący *US* steruje pozostałymi blokami funkcjonalnymi systemu i realizuje komunikację z mikrokomputerem IBM PC. Jest wyposażony w wejście i wyjście równoległe, przeznaczone do współpracy z programatorem pamięci stałych.

System SUMUS2 zawiera machanizmy samotestujące. Zawsze podczas inicjowania systemu jest wykonywany test sprawności wszystkich bloków części sprzętowej. Operator może też w każdej chwili wykonać test sond, umieszczając je w specjalnych złączach diagnostycznych.

Oprogramowanie i możliwości funkcjonalne MIDES–a

Na stałe oprogramowanie systemu uruchomieniowego MIDES składają się dwa programy (rys. 2):
- RES – oprogramowanie rezydentne inteligentnego terminala SUMUS2,
- SUM – główny program sterujący pracą całej stacji i realizujący komunikację z operatorem przez konsolę komputera IBM PC.



Rys. 2. System MIDES

W programie *RES* zaimplementowano procedury gwarantujące właściwe sterowanie sprzętem (SUMUS–em). Główne zadania programu SUM są następujące:
- udostępnienie użytkownikowi możliwie najlepszego zestawu funkcji przeznaczonych do uruchamiania urządzeń mikroprogramowanych; podstawowym wymaganiem jest tu mksymalne wykorzystanie możliwości sprzętowych systemu SUMUS,
- zapewnienie sprawnej i niezawodnej komunikacji z SUMUS–em,
- zorganizowanie i obsługa wygodnego podsystemu komunikacji z operatorem,
- zagwarantowanie możliwości współpracy z innymi programami, np. metaasemblerem lub urządzeniami (np. emulatorem sekwensera 2910).

W obecnej wersji w skład oprogramowania stanowiska roboczego MIDES oprócz programów SUM i RES wchodzi także metaasembler MIC. Programy pracujące na IBM PC (tzn. SUM i MIC) zostały tak napisane, że ich działanie nie zależy od konfiguracji mikrokomputera. Programy te pracują pod kontrolą systemu PC–DOS o numerze wersji 2.0 lub wyższym.

LITERATURA

[1] Pawłowski M.: Mikroprogramowanie. Informatyka, nr 4, 5, 1987.
[2] Small C.H.: Development system for 32–bit μPs. END, June 13 1985, pp. 107–122.

CRAY-2

dokończenie ze s. 20

W konfiguracji VAX-11 istniał też koprocesor FPA (ang. *Floating Point Accelerator*). Na maszynach IBM i VAX testy przeprowadzono dla pojedynczej (32 bity) i podwójnej (64 bity) precyzji liczb zmiennoprzecinkowych. Na maszynach Cray przetestowano ten sam kod w wersji skalarnej (bez wektoryzacji) i wektorowej (z wektoryzacją). W tabeli 2 zaprezentowano wyniki wybranych testów.

• • •

Na początku 1988 r. firma Cray Research przedstawiła nowy model, Cray Y-MP, następcę Cray X-MP. Zbudowano go z zaprojektowanych samodzielnie przez firmę układów VLSI (dotychczas korzystano z dostępnych na rynku elementów firm Fairchild, Motorola i in.). Cray Y-MP ma osiem procesorów; czas cyklu zegara skrócono do 6 ns. Kontynuowana jest również linia CRAY-2, w postaci zapowiadanego na przełom 1988/1989 systemu Cray-3, będącego implementacją Cray-2 w technologii arsenku galu, ze zwiększeniem liczby procesorów do 16. Czas cyklu zegara ma zostać skrócony do ok. 2 ns (a według niektórych źródeł nawet do 1 ns), co daje moc obliczeniową co najmniej 1 GFLOPS dla jednego procesora.

LITERATURA

[1] Bucher I. Y., Simmons M. L.: Performance Assessment of Supercomputers. Report LA-UR-85-1505, Los Alamos National Laboratory
[2] Hockney R. W.: Parallel Computers – Architecture and Performance. Parallel Computing 85, North Holland, Amsterdam, 1986
[3] Hwang K., Briggs F. A.: Computer Architecture and Parallel Processing. McGraw-Hill, New York, 1984
[4] Iszkowski W.: Superkomputer Cray. Informatyka, nr 5, 1985
[5] Lubecki O., Moore J., Mendez R.: A Benchmark Comparison of Three Supercomputers – Fujitsu VP-200, Hitachi S810/20, and Cray X-MP/2. Proc. 1st. Intern. Conf. on Supercomputing Systems 1985. North-Holland, Amsterdam, 1986
[6] Solid-State Storage Device, Cray Research, Inc., 1987
[7] The Cray-2 Computer System. Cray Research, Inc., 1986
[8] The Cray X-MP Series of Computer Systems. Cray Research Inc., 1987
[9] Thompson J. R.: The Cray-1, the Cray X-MP, the Cray-2 and Beyond – The Supercomputers of Cray Research. Supercomputers. North-Holland, Amsterdam, 1986

JACEK KURAŚ
Centrum Podstawowych Problemów Gospodarki Surowcami Mineralnymi i Energią
Polska Akademia Nauk
Kraków

JACEK LEMBAS
MAREK SKOMOROWSKI
Katedra Informatyki
Uniwersytet Jagielloński
Kraków

Język symulacji systemów o działaniu ciągłym ze zdarzeniami dyskretnymi

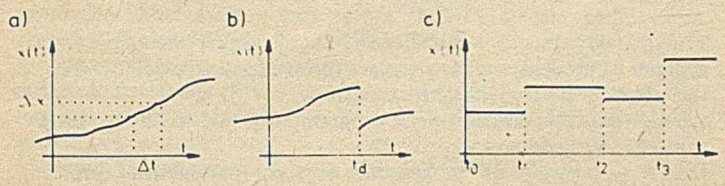
Jedną z podstawowych właściwości systemu, którą bada się w procesie symulacji, jest jego działanie, czyli zachowanie się zmiennych stanu w czasie. Codzienne odczucie czasu fizycznego nakazuje traktować go jako zmienną ciągłą o wartościach należących do zbioru liczb rzeczywistych. Takie traktowanie czasu jest słuszne dla makroskopowego ujęcia zjawisk w otaczającym nas świecie i przestaje być słuszne dopiero na poziomie zjawisk, którymi zajmuje się fizyka kwantowa. Istnieje jednak wiele systemów, których działanie ma taki charakter, że zasadnicze, istotne zmiany stanu zachodzą w nich jedynie w pewnych dyskretnych momentach czasu, między którymi wartości zmiennych stanu nie zmieniają się lub zmieniają się bardzo niewiele. Przy tworzeniu modeli takich systemów posługiwanie się ciągłą zmienną czasową wydaje się nieuzasadnione. Tak więc, z punktu widzenia działania, systemy możemy podzielić na dwie klasy:

- systemy o działaniu ciągłym, w których w każdym, dowolnie małym odcinku czasu zachodzą zmiany stanu mające istotny wpływ na dalsze zachowanie systemu, a ponadto – małym zmianom czasu odpowiadają małe zmiany stanu,
- systemy o działaniu dyskretnym, w których zmiany stanu, istotne z punktu widzenia celu badania systemu, zachodzą w dyskretnym (przeliczalnym) zbiorze chwil czasowych.

Na pograniczu tych dwu klas można wyróżnić jeszcze trzecią:

- systemy o działaniu ciągłym ze zdarzeniami dyskretnymi, w których zasadniczo ciągłe działanie systemu jest zaburzane w dyskretnych chwilach czasowych przez zjawiska powodujące skokowe zmiany stanu.

Ilustrację powyższego podziału stanowi rys. 1.



Rys. 1. Ilustracja zachowania pewnej znaczonej stanu x(t): a) w systemie o działaniu ciągłym, b) w systemie o działaniu ciągłym ze zdarzeniami dyskretnymi, c) w systemie o działaniu dyskretnym

Języki do symulacji komputerowej systemów o działaniu ciągłym bazowały początkowo na tradycji modelowania analogowego i z tego powodu nazwano je cyfrowymi symulatorami maszyn analogowych. Do tej grupy języków należą m.in. język 1130/CSMP [1] oraz GODYS–1 i 2 [2, 4]. W latach 1965–1967 powstał język CSSL (ang. *Continuous Systems Simulation Language)*, uznany później za wzorcowy język do symulacji systemów o działaniu ciągłym. Autorzy tego języka odeszli w znacznym stopniu od tradycji modelowania analogowego, pozostawiając użytkownikowi znaczną swobodę w doborze formy zapisu równań modelu. W ogólności, forma ta ma postać równań podobnych syntaktycznie do instrukcji przypisania w językach programowania. Translator języka CSSL–3, będącego pewną wersją języka CSSL w systemie Cyber–72, został skonstruowany jako makroprocesor. Tłumaczy on program w języku CSSL–3 na program w języku Fortran, który jest następnie kompilowany i ładowany do pamięci z użyciem tradycyjnych środków. Zaletą takiego rozwiązania jest możliwość stosowania makrodefinicji i różnego rodzaju bibliotek. Rozwiązanie to charakteryzuje się jednak bardzo długim czasem translacji. Ponadto diagnostyka błędów na etapie wykonania jest szczególnie trudna, ponieważ standardowe procedury diagnostyczne odwołują się do programu pośredniego zapisanego w Fortranie, a nie do programu źródłowego napisanego przez użytkownika w języku CSSL–3.

Językami zbliżonymi zewnętrznie do CSSL–3 są języki GODYS–5 [5, 6] i GODYS–6 [8] zaimplementowane na komputerach serii Odra i Riad oraz przedstawiony w niniejszym opracowaniu język GODYS–PC, zrealizowany na mikrokomputery kompatybilne z IBM PC/XT/AT. Języki GODYS–5 i 6 jak również język GODYS–PC zostały zaprojektowane w taki sposób, że nie posiadają wymienionych wyżej wad języka CSSL–3.

Popularność i dostępność mikrokomputerów zgodnych z IBM PC spowodowała, że wiele obliczeń wykonywanych dotychczas na dużych komputerach realizuje się na mikrokomputerach. W bogatej ofercie oprogramowania mikrokomputerów IBM PC odczuwa się brak języków służących do symulacji zarówno procesów o działaniu ciągłym, jak i dyskretnym. Spotyka się rzadko zarówno języki przeniesione wprost z dużych komputerów (np. Dynamo), jak też uwzględniające specyfikę pracy z mikrokomputerem (np. ISIM firmy Simulation Sciences).

Należy nadmienić, że procesy symulacyjne są procesami czasochłonnymi i symulacja wielkich złożonych modeli na mikrokomputerach 16-bitowych może okazać się nieopłacalna lub wprost niemożliwa. Symulacja małych i średnich modeli z wykorzystaniem mikrokomputera jest możliwa, a częstokroć bardziej elastyczna niż w wypadku dużych komputerów.

Badania symulacyjne przeprowadzone porównawczo na identycznych modelach na komputerze Riad–32 (GODYS–6) i na mikrokomputerze zgodnym z IBM PC/AT z zegarem 10 MHz (GODYS–PC) wykazały 15% wzrost szybkości obliczeń na korzyść mikrokomputera. Powyższe uwagi uzasadniają celowość wprowadzenia języka GODYS–PC do oprogramowania mikrokomputerów kompatybilnych z IBM PC.

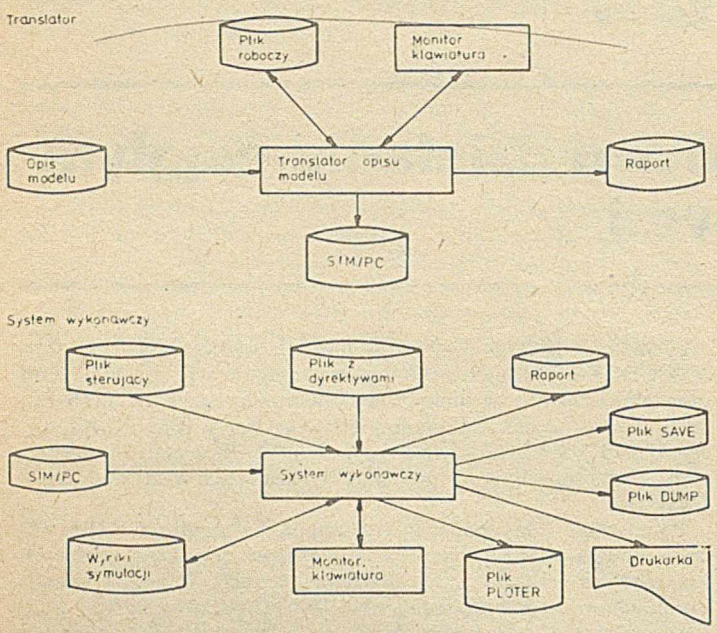
ZAŁOŻENIA I MOŻLIWOŚCI JĘZYKA GODYS–PC

Modelem matematycznym systemu o działaniu ciągłym jest, w najprostszym przypadku, układ równań różniczkowych zwyczajnych. Jeżeli układ ten jest podany w postaci kanonicznej, to nadaje się bezpośrednio do całkowania numerycznego. W praktyce trudno jednak żądać od użytkownika sprowadzania jego modelu do postaci kanonicznej, ponieważ ta forma może być dla niego niedogodna i nienaturalna. Język GODYS–PC pozwala na zapisanie modelu użytkownika w formie równań o praktycznie dowolnej postaci. Numeryczne metody całkowania równań różniczkowych zwyczajnych wymagają jednak przeprowadzania obliczeń w pewnej standardowej kolejności (tzw. scentralizowane całkowanie). Powstaje więc problem ustalenia tej kolejności w modelu użytkownika. Zadanie to musi wykonać translator języka symulacyjnego. Do ustalenia prawidłowej kolejności obliczeń w języku GODYS–PC wykorzystano formalną metodę, opartą na grafach funkcyjnych, przedstawioną w [3].

Podstawą przygotowania programu w języku GODYS–PC jest zapis modelu matematycznego badanego systemu w postaci równań różniczkowo-całkowo-logicznych. Zasadniczą częścią programu jest opis tego modelu, odwzorowany w procesie translacji na model cyfrowy. Na modelu tym można dokonywać wielu eksperymentów symulacyjnych. Instrukcje opisu modelu mogą być pisane w dowolnej kolejności, ponieważ translator języka GODYS–PC dokonuje ich automatycznego sortowania zapewniając właściwą kolejność obliczeń. Algorytm wyznaczania sekwencji obliczalnej (tj. określenia kolejności wykonywania instrukcji) został przedstawiony w [7].

Do całkowania równań modelu można wykorzystać jedną z sześciu dostępnych w języku metod całkowania: pięć metod stałokrokowych

i jedną zmiennokrokową. Wyniki eksperymentów mogą być prezentowane w postaci odpowiednich tabel i wykresów. Procesor języka GODYS–PC stanowią dwa programy: translator języka opisu modelu i system wykonawczy (rys. 2).



Rys. 2. Struktura systemu GODYS-PC

Procesor ten można wykorzystywać zarówno w tradycyjnym trybie wsadowym, jak i w trybie konwersacyjnym. Kompilator języka opisu modelu analizuje program wcześniej zapisany w całości na pliku. Nie ma więc znaczenia, w jaki sposób jest wprowadzany jego tekst. Różnica jakościowa występuje przy wsadowym i konwersacyjnym trybie wykorzystania systemu wykonawczego. W trybie wsadowym należy przewidzieć (przynajmniej w zarysie) wyniki poszczególnych dyrektyw, aby następne dyrektywy były sensowne. Tryb konwersacyjny daje możliwość bezpośrednich zmian w doborze kolejnych dyrektyw symulacji w zależności od dotychczasowych wyników. Jest to szczególnie ważne przy przypadkowym dobieraniu aktualnych parametrów kolejnego eksperymentu symulacyjnego. Można również prześledzić na monitorze kilka wersji dyrektywy wydawniczej, a do składania i wydruku wybrać tę najlepszą. Trzeba ponadto zaznaczyć, że dyrektywy te korzystają z procedur graficznych, czego brak było w poprzednich wersjach języka GODYS, a co znacznie polepsza jakość systemu. Sposób użytkowania systemu wykonawczego w trybie konwersacyjnym jest prosty, zaopatrzony w podstawowe mechanizmy przerwań i nie wymaga dodatkowej instrukcji.

STRUKTURA PROGRAMU W JĘZYKU GODYS–PC

Program w języku GODYS–PC składa się z dwu zasadniczych części:
– zapisu modelu matematycznego systemu,
– sekwencji dyrektyw sterujących eksperymentami symulacyjnymi na modelu i wyprowadzaniem wyników.

W procesie translacji zapis modelu symulowanego systemu jest przekształcany na równoważny program w języku maszyny abstrakcyjnej SIM/PC (rys. 2), implementowanej przez efektywny interpreter, którego pracą sterują wymienione wyżej dyrektywy.

Program w języku GODYS–PC ma następującą strukturę:

```
MODEL nazwa-modelu
    sekcja deklaracji (PARAMT, CONST, PREPAR, DECLARE,
                       MAP, NOMAP, RENAME)
INITIAL
    sekcja inicjująca (opcjonalnie)
END
DYNAMIC
    instrukcje opisu modelu
END
LOAD nazwa-modelu
    ciąg dyrektyw (DATA, EXECUTE, CONTINUE, HDR, PRINT,
                   PRPLOT, GRAPH, PLOTXY, DUMP, SAVE)
FINISH
```

Dyrektywa MODEL podaje symboliczną nazwę modelu. Po dyrektywie MODEL następuje sekcja deklaracji, w której mogą występować następujące deklaracje:

PARAMT –deklaruje identyfikatory parametrów modelu. Parametrom tym, w trakcie kolejnych eksperymentów, można nadawać wartości (za pomocą dyrektywy DATA), odwołując się do nich przez nazwę. Za pomocą dyrektywy PARAMT deklarowane są parametry proste i tablice jednowskaźnikowe.

CONST – definiuje identyfikatory stałych liczbowych modelu.

PREPAR – deklaruje zmienne obserwowane modelu, tzn. zmienne rejestrowane w trakcie eksperymentów symulacyjnych. Zmienna niezależna, o standardowym identyfikatorze T, jest rejestrowana automatycznie.

DECLARE – określa liczbę argumentów i parametrów funkcji zdefiniowanej przez użytkownika.

MAP – deklaracja ta umożliwia wydruk pełnej mapy modelu (standardowo wyprowadzana jest tylko mapa identyfikatorów).

NOMAP – deklaracja ta wstrzymuje wyprowadzanie mapy.

RENAME – deklaracja ta zmienia identyfikator zmiennej niezależnej modelu (standardowo T). Na przykład, użycie RENAME T = CZAS powoduje, że nazwą zmiennej niezależnej będzie CZAS, a identyfikator T może być użyty do oznaczenia innej zmiennej.

Sekcja inicjująca ujęta w ograniczniki INITIAL – END powoduje nadanie wartości parametrom wewnętrznym modelu. Identyfikatory występujące w sekcji inicjującej po lewej stronie znaku przypisania są identyfikatorami parametrów wewnętrznych. Parametrom tym przypisywane są odpowiednie wartości przed rozpoczęciem eksperymentu symulacyjnego.

Właściwy opis modelu tworzy sekwencja instrukcji opisu modelu ujęta w ograniczniki DYNAMIC – END. Instrukcja opisu modelu ma postać standardowej instrukcji przypisania, przy czym zmienna występująca po lewej stronie znaku przypisania może wystąpić w tym miejscu tylko jeden raz. Wyrażenie występujące po prawej stronie znaku przypisania jest konstruowane przy użyciu operatorów arytmetycznych, relacyjnych, logicznych i funkcji standardowych języka. Funkcje standardowe języka mają ściśle określone zbiory argumentów i parametrów. Parametrami aktualnymi funkcji standardowych mogą być jedynie wielkości stałe w trakcie danego eksperymentu symulacyjnego (stałe, nazwy stałych, parametry wewnętrzne, parametry proste i tablice). Język GODYS–PC ma 75 operatorów i funkcji standardowych. Użytkownik może w bardzo prosty sposób rozszerzyć język GODYS–PC o własne funkcje napisane w języku Fortran–77.

Dyrektywy systemu wykonawczego sterują eksperymentami symulacyjnymi na modelu, a także sterują wyprowadzaniem wyników. Wśród tych dyrektyw wyróżnia się następujące:

LOAD – powoduje załadowanie programu w języku maszyny abstrakcyjnej SIM/PC do pamięci.

DATA – pozwala na przypisanie wartości parametrom prostym, a tablicom zbiorów wartości. Wartości te mogą być zmienione tylko kolejną dyrektywą DATA, lub w trakcie procesu optymalizacji parametrycznej.

EXECUTE, CONTINUE – powodują wykonanie pojedyńczego eksperymentu symulacyjnego na modelu o określonych aktualnie parametrach. Eksperyment jest autonomiczny w wypadku EXECUTE lub jest kontynuacją eksperymentu poprzedniego w wypadku CONTINUE. Sposób prowadzenia eksperymentu symulacyjnego determinują parametry wykonania. Parametry te określają odpowiednio:
TMIN, TMAX – modelowy czas początku i końca symulacji,
METHOD – algorytm całkowania numerycznego,
DT – krok całkowania numerycznego,
ABSERR, RELERR – tolerancję dla bezwzględnego i względnego błędu całkowania w metodzie zmiennokrokowej,
COMDEL – przyrost zmiennej niezależnej (okres komunikacyjny), determinujący częstotliwość zapisywania wartości zmiennych obserwowanych na plik roboczy,
BEGTR, ENDTR – modelowy czas początku i końca śledzenia zmiennych modelu,
CLKTIME – limit rzeczywistego czasu trwania eksperymentu symulacyjnego (w sekundach),

OPT = (lista) – optymalizacja parametryczna (dozwolona tylko w dyrektywie EXECUTE); lista parametrów optymalizacji zawiera identyfikator funkcji celu jako przyjęty wskaźnik jakości, identyfikatory optymalizowanych parametrów (maksymalnie 8) wraz z zakresem ich występo-

wania oraz informacje o niestandardowej obsłudze zakończenia optymalizacji. Wystąpienie błędu wykonania w danym eksperymencie powoduje jego zakończenie. Kolejne eksperymenty mogą być realizowane tylko w wypadku podania parametru wymuszającego wykonanie FORCE. W wypadku niepodania wszystkich lub tylko pewnych parametrów w dyrektywach EXECUTE lub CONTINUE przypisywane są im wartości standardowe lub zastępcze, implikowane przez inne, podane wcześniej przez użytkownika.

Wygodnym elementem języka są dyrektywy wydawnicze. Umożliwiają one prezentowanie na monitorze, drukarce, pliku tekstowym lub ploterze zmiennych obserwowanych modelu, rejestrowanych w trakcie eksperymentu symulacyjnego, w postaci:
PRINT – tablica wartości,
PRPLOT – wykresy (do 4 zmiennych równocześnie),
GRAPH – tablica wartości i wykres dla jednej zmiennej,
PLOTXY – wykres zależności dwóch zmiennych.
Przy wykonywaniu wykresów wykorzystywana jest skala określana automatycznie lub też ustalana na podstawie podanych przez użytkownika wartości granicznych danej zmiennej. Inne dyrektywy są następujące:
HDR – podaje nagłówek,
SAVE – powoduje składowanie wyników symulacji (wartości zmiennych obserwowanych) na pliku, który może być później wykorzystany przez inne programy do obliczeń postsymulacyjnych.
DUMP – powoduje kopiowanie na plik zawartości pamięci maszyny abstrakcyjnej SIM/PC.
FINISH – powoduje zakończenie programu.

Język GODYS–PC umożliwia pracę zarówno w tradycyjnym trybie wsadowym, co jest szczególnie pożyteczne w wypadku dłuższych obliczeń, jak również w trybie konwersacyjnym, pozwalającym na bieżące śledzenie procesu symulacyjnego i korygowanie parametrów na podstawie obserwacji.

PRZYKŁADOWY PROGRAM

Jako przykład programu w języku GODYS–PC rozważmy następujący model sterowania zapasami [6]. Detalista utrzymuje zwykle pewien poziom zapasów sprzedawanych towarów, który ustala jako kompromis między kosztem składowania a stratami, jakie mogłyby być poniesione wskutek braku towaru. Różnica między żądanym poziomem zapasów (ZPZ) a bieżącym poziomem zapasów (BPZ) jest wykorzystywana do sterowania (system ze sprzężeniem zwrotnym) natężeniem zamówień składanych u producenta (NZSP) w celu uzupełnienia zapasów. Bieżące natężenie produkcji (BNP) jest proporcjonalne do liczby zamówień nie zrealizowanych przez producenta (ZNPP). Bieżące natężenie sprzedaży (BNS), będące zmienną zewnętrzną modelu (wymuszeniem) ma postać BNS = $f(t)$, gdzie $f(t)$ jest dowolną funkcją spełniającą warunek $f(t) \geq 0$ (warunek ten wynika z fizycznej interpretacji bieżącego natężenia sprzedaży.

W rozważanym modelu zmiennymi stanu są następujące zmienne: bieżący poziom zapasów (BPZ), zamówienia nie zrealizowane przez producenta (ZNPP) i średnie natężenie sprzedaży (*SNS*), dla których równania stanu przyjmują postać:

$$\frac{d(\mathrm{BPZ})}{dt} = \mathrm{BNP} - \mathrm{BNS}$$

$$\frac{d(\mathrm{ZNPP})}{dt} = \mathrm{NZSP} - \mathrm{BNP}$$

$$\frac{d(\mathrm{SNS})}{dt} = \frac{\mathrm{BNS} - \mathrm{SNS}}{T2}$$

$T2$ reprezentuje jednostkę czasu, w jakiej następuje uśrednianie sprzedaży. Program w języku GODYS-PC dla opisanego modelu przedstawiono na rys. 3.

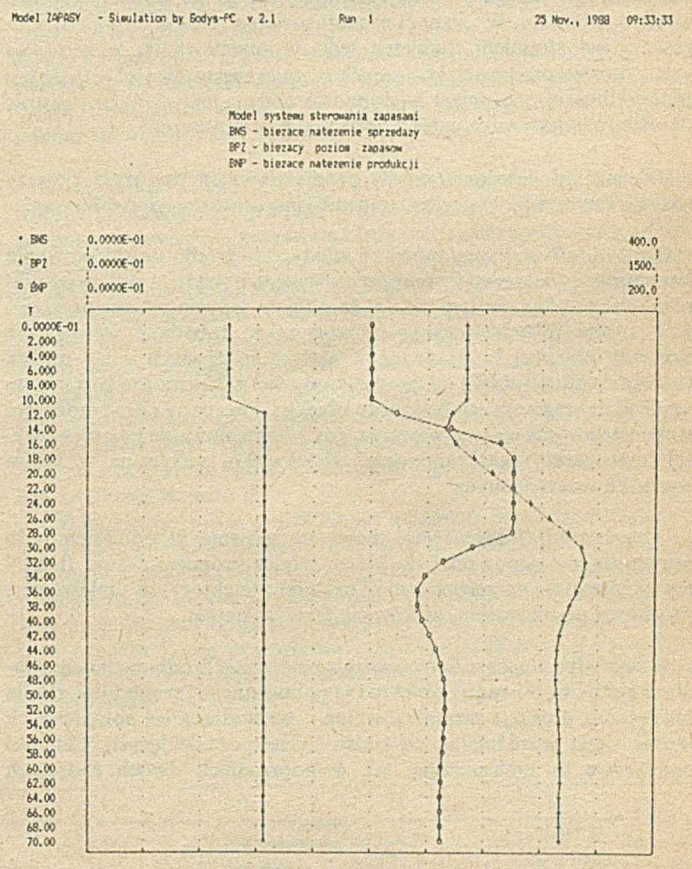
W rozważanym modelu przyjęto pewne ograniczenia. Bieżący poziom zapasów (BPZ) nie może przyjmować wartości ujemnych. Wynika to z fizycznej interpretacji zapasów. Przyjęto również, że bieżące natężenie produkcji (BNP) może wzrastać tylko do pewnej wartości maksymalnej. Ograniczenia te zostały zrealizowane za pomocą funkcji standardowych języka GODYS-PC (MAX, BOUND). W przedstawionym programie bieżące natężenie sprzedaży (BNS) ma postać funkcji

```
model zapasy
nosap
 prepar bns, bpz, bnp
 paramt t1, t2, t3, k1, k2, k3
 paramt bpz0, znpp0, sns0, r, a1, a2, t0, b
dynamic
   z = integ(bnp-bns;bpz0)
   bpz = max(z,0)
   znpp = integ(nzsp-bnp;znpp0)
   sns = integ((bns-sns)/t2;sns0)
   bnp = k3*bound(znpp;1,b)/t1
   zpz = r*sns
   nzsp = k1*sns+k2*(zpz-bpz)/t3
   bns = a1*step(t)+a2*step(t-t0)
end

load zapasy
data t1=4,t2=4,t3=4,k1=1,k2=1,k3=1,bpz0=1000,znpp0=400.
     sns0=100,r=10,a1=100,a2=25,t0=10,b=600
hdr "              Model systemu sterowania zapasami"
hdr "              BNS - biezace natezenie sprzedazy"
hdr "              BPZ - biezacy  poziom  zapasow"
hdr "              BNP - biezace natezenie produkcji"
execute(dt=0.2,tmax=70,comdel=2,method=trapez)
prplot(bns=(0,400),bpz=(0,1500),bnp=(0,200))
finish
```

Rys. 3. Program symulacyjny w języku GODYS-PC dla modelu systemu sterowania zapasami

skoku jednostkowego (funkcja standardowa STEP). Wyniki eksperymentu symulacyjnego dla przedstawionego modelu systemu sterowania zapasami zostały przedstawione na rys. 4.



Rys. 4. Wyniki eksperymentu symulacyjnego dla modelu systemu sterowania zapasami; realizacja dyrektywy PRPLOT z rys. 3

* * *

Język GODYS-PC jest eksploatowany od jesieni 1987 r. zarówno do celów badawczych, jak i dydaktycznych. Dotychczasowe doświadczenia uzyskane w trakcie eksploatacji języków GODYS-5 i GODYS-6, a także języka GODYS-PC pozwalają twierdzić, że GODYS-PC jest prostym i wygodnym narzędziem do symulacji szerokiej klasy systemów dynamicznych o działaniu ciągłym, jak również systemów o działaniu ciągłym ze zdarzeniami dyskretnymi. Szczególnym walorem odróżniającym go od poprzedników są wszechstronne możliwości wydawnicze, a także możliwość śledzenia na bieżąco eksperymentów symulacyjnych przez użytkownika i reagowanie w wypadku podania błędnych parametrów (zerwanie symulacji, zakończenie zbyt długiego procesu wydawniczego).

dokończenie na s. 3

MARZENA TATAJ
ALINA TORUŃ
Instytut Podstaw Informatyki
Polska Akademia Nauk
Warszawa

Tomografia komputerowa

Termin *tomografia wspomagana komputerowo* lub *tomografia komputerowa* wszedł do powszechnego użycia w latach siedemdziesiątych, po zbudowaniu pierwszego tomografu komputerowego przez Godfreya Hounsfielda w 1972 r. (nagroda Nobla z 1979 r.).

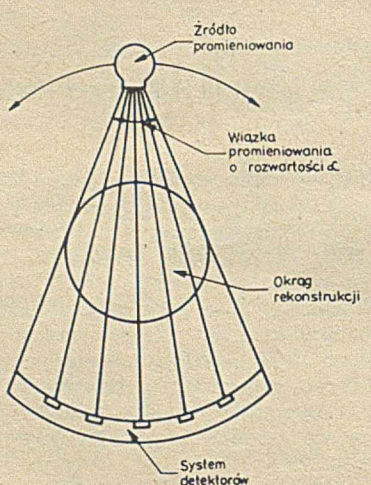
Tomografia komputerowa to technika, za pomocą której można uzyskać obraz wnętrza badanego obiektu, ściślej rozkład *gęstości* cech wewnątrz obiektu, na podstawie zebranych danych pomiarowych. *Gęstość* oznacza w tym wypadku przestrzenny rozkład wybranej cechy rekonstruowanego obiektu. Jest ona różnie interpretowana w zależności od rodzaju fizycznego procesu wykorzystywanego do uzyskania obrazu wnętrza obiektu. W wypadku zastosowania promieniowania X – gęstość jest stopniem tłumienia tego promieniowania, w wypadku zastosowania prądów elektrycznych o małej częstotliwości gęstość jest przewodnością, natomiast w wypadku pól magnetycznych – gęstość określa rozmieszczenie jąder wodorowych w organizmie.

Zależnie od położenia źródła promieniowania rozróżnia się trzy rodzaje tomografii: emisyjną, transmisyjną i transmisyjno-emisyjną.

O *tomografii emisyjnej* mówi się wtedy, gdy źródło promieniowania jest umieszczone wewnątrz rekonstruowanego obiektu, a system pomiarowy jest pasywny w tym sensie, że odbiera jedynie promieniowanie. Jeśli źródła promieniowania (a więc także i badane wewnętrzne elementy obiektu) są osadzone w gęstym środowisku – nie można rozróżnić gęstości źródła od gęstości środowiska. Nie można tym samym uzyskać dobrego obrazu badanego obiektu. Pożądany obraz może być zrekonstruowany jedynie wówczas, gdy promieniowanie jest, co najwyżej, nieznacznie zakłócane przez środowisko, nazywane w takim wypadku rozrzedzonym.

O *tomografii transmisyjnej* mówi się głównie w odniesieniu do tomografii rentgenowskiej, w której źródła promieniowania (lampy rentgenowskie) i sprzężone z nimi czujniki (detektory) są umieszczone na okręgu pomiarowym, całkowicie poza obiektem.

W wypadku *tomografii transmisyjno-emisyjnej* źródła promieniowania znajdują się wewnątrz obiektu i są pobudzane przez specjalne źródła zewnętrzne, a emisja promieniowania – pochodząca od pobudzanych źródeł – jest rozróżnialna od emisji źródeł pobudzających. Czujniki pomiarowe są umieszczone, jak w poprzednich dwóch rodzajach tomografii, na zewnątrz obiektu. W tomografii transmisyjno-emisyjnej wykorzystywane jest zjawisko jądrowego rezonansu magnetycznego (JRM).



Rys. 1. Zasada zbierania danych pomiarowych w tomografii transmisyjnej

W każdym rodzaju tomografii badany obiekt trójwymiarowy jest umieszczony wewnątrz okręgu zwanego *okręgiem rekonstrukcji* (rys. 1). Okrąg ten wyznacza obszar, z którego zbierane są dane pomiarowe (podlegający rekonstruowaniu). Środowisko wypełniające przestrzeń poza okręgiem jest uznawane za rozrzedzone (tzn. zakłada się, że jego gęstość jest pomijalna); przyjmuje się bowiem założenie, że emisja promieniowania zachodzi jedynie między rekonstruowanym obiektem i systemem pomiarowym. W rzeczywistości część struktury podtrzymującej obiekt również przechwytuje promieniowanie i jest traktowana jako fragment badanego obiektu.

Oprócz okręgu rekonstrukcji otaczającego badany obiekt można zdefiniować również okrąg pomiarowy (współśrodkowy z poprzednim), na którym umieszczone są detektory odbierające emitowane promieniowanie oraz – w wypadku tomografii transmisyjnej – źródła promieniowania.

Głównym miernikiem przydatności danego rodzaju tomografii do poszczególnych zastosowań jest *rozdzielczość* rekonstruowanego obrazu. Zależy ona głównie od rozmiarów detektorów wyznaczających wymiary pojedynczego elementu rekonstruowanego obiektu tzw. *woksela* (ang. *voxel*). Jest to związane zazwyczaj z obszarem naświetlania, czyli rozwartością kątową wiązki promieniowania.

TOMOGRAFIA RENTGENOWSKA

Potocznie terminem *tomografia komputerowa* określa się metodę badań rentgenologicznych, polegającą na wykonywaniu warstwowych zdjęć (poprzecznych przekrojów) struktury wewnętrznej za pomocą specjalistycznej aparatury rentgenologicznej. Wizualizacja tego przekroju – *plastra* badanego obiektu – dostarcza wielu cennych informacji o stanie jego wnętrza. Gęstość G każdego punktu plastra mierzy się w skali Hounsfielda, według zależności:

$$G = \frac{\mu - \mu_{H20}}{\mu_{H20}} \cdot 1000 \quad [\mathrm{H}]$$

gdzie H jest jednostką skali Hounsfielda, μ – liniowym współczynnikiem pochłaniania promieniowania X dla danego materiału i ustalonego poziomu energetycznego, μ_{H20} – liniowym współczynnikiem pochłaniania promieniowania dla wody i ustalonego poziomu energetycznego. Tak więc skala Hounsfielda ma 2000 dyskretnych poziomów, z których charakterystyczne są trzy następujące: 0[H] – woda, 1000[H] – kość, 1000[H] – powietrze.

W wypadku tomografii rentgenowskiej badany obiekt jest umieszczony wewnątrz okręgu rekonstrukcji (rys. 1). Źródłem promieniowania jest lampa rentgenowska trwale połączona z systemem detektorów rozmieszczonych wewnątrz kolistej ramy. Obiekt jest naświetlony wachlarzową wiązką promieniowania o rozwartości α. Natężenie promieniowania rentgenowskiego, po przejściu przez rekonstruowany obiekt, mierzone jest przez system detektorów. Pomiar natężenia promieniowania, dla każdego kątowego położenia lampy, jest nazywany *projekcją*, *skanem* lub *omiotem*. Jedna projekcja – to zbiór wartości natężenia wiązki dla danego ustawienia lampy rentgenowskiej (mniejszych lub równych natężeniu wiązki emitowanej przez lampę). Osłabienie wiązki promieniowania zależy od rodzaju obiektu znajdującego się na drodze promieni X. W wypadku zastosowania 180 detektorów,

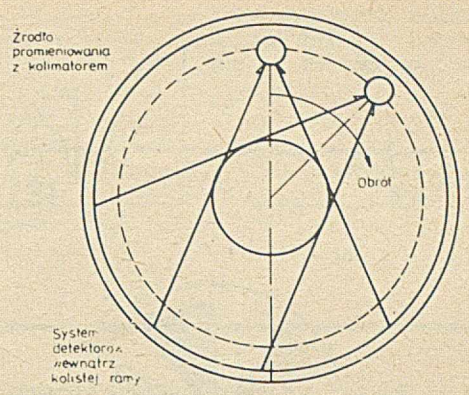
rejestrujących natężenie wiązki promieniowania po przejściu przez obiekt, przy jednym położeniu lampy uzyskuje się 180 wartości dla jednej projekcji. Po dokonaniu pełnego obiegu źródła wokół obiektu otrzymuje się komplet danych pomiarowych.

W tomografii rentgenowskiej można rozróżnić następujące etapy obróbki pomiarów, które prowadzą do uzyskania odpowiedniej postaci danych wejściowych dla algorytmów rekonstruowania:
- zbieranie danych z detektorów,
- logarytmowanie,
- przetwarzanie analogowo-cyfrowe,
- wstępne przetwarzanie, korekcja błędów wynikających z niedoskonałości elementów mechanicznych i elektrycznych urządzeń pomiarowych.

Po ostatnim etapie otrzymuje się zbiór wartości, reprezentujących sumy pochłaniania promieniowania X wzdłuż poszczególnych promieni, równy liczbowo iloczynowi liczby projekcji i liczby detektorów. Dane te podlegają następnie dalszej, matematycznej obróbce według algorytmów rekonstruowania. W końcowej fazie przetworzone dane mają postać obrazu rozkładu wartości pochłaniania promieniowania X w badanym obszarze.

Tak więc system tomograficzny spełnia cztery podstawowe funkcje:
- zbieranie danych pomiarowych,
- wstępne przetwarzanie danych pomiarowych,
- rekonstruowanie,
- wizualizacja i analiza zrekonstruowanego obrazu.

Budowa systemów *zbierania danych pomiarowych* jest zróżnicowana w zależności od struktury poszczególnych typów tomografów. Pierwsze systemy tomografii komputerowej zostały uruchomione w 1972 r. Były one przeznaczone wyłącznie do badania głowy, ze względu na długi czas zbierania danych i małe pole pomiarowe. Skaner tomografów I generacji ma jedno źródło emitujące, tzw. wiązkę ołówkową promieniowania X i sprzężony z nim detektor. Dla każdego kątowego położenia źródła, zarówno lampa, jak i detektor, przesuwają się liniowo (względem siebie równolegle, prostopadle do promieni wiązki), co umożliwia zebranie projekcji dla promieni równoległych. W wypadku tomografów II generacji, przeznaczonych do badania całego ciała, układ pomiarowy zawiera zamiast jednego detektora – tablicę detektorów przesuwnych, co wynika z zastosowania wąskiej wiązki wachlarzowej (o rozwartości od 5° do 12°). Wykorzystanie tablicy detektorów i wachlarzowej wiązki promieniowania umożliwia przyspieszenie ruchu liniowego układu pomiarowego, a przez to zmniejszenie czasu wykonywania pomiarów (do kilkudziesięciu sekund). W tomografach następnych generacji stosowano już wyłącznie źródła promieniowania emitujące szeroką wiązkę wachlarzową i odpowiadające jej systemy detektorów, rozmieszczonych wewnątrz kolistej ramy. W wypadku systemów IV generacji (rys. 2) lampa rentgenowska nie jest sprzężona z systemem detektorów, lecz obraca się w jego wnętrzu. Pojedyncza projekcja zawiera zatem dane pomiarowe ze wszystkich detektorów objętych wiązką promieniowania. Tabela prezentuje krótką charakterystykę kolejnych generacji tomografów rentgenowskich do zastosowań medycznych ze szczególnym uwzględnieniem budowy ich skanerów.



Rys. 2. System pomiarowy tomografu IV generacji

TOMOGRAFIA STRUKTUR DYNAMICZNYCH

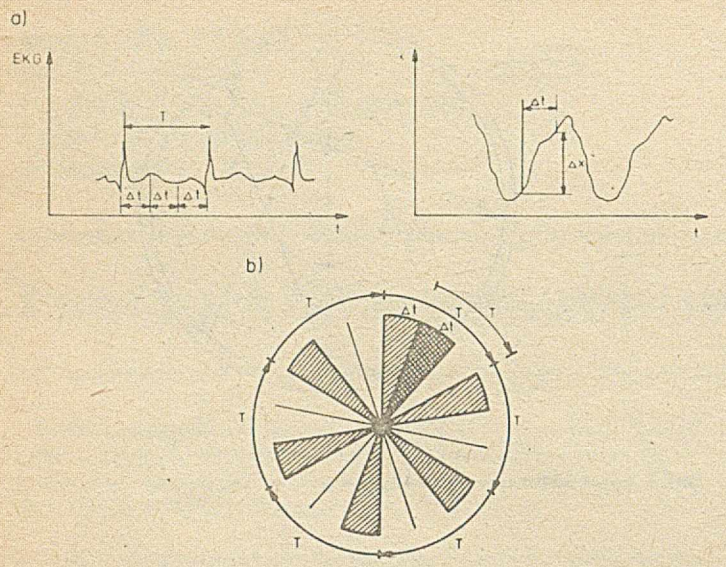
Obrazy statyczne, uzyskiwane za pomocą konwencjonalnych systemów tomograficznych, umożliwiają określenie struktury (anatomii) badanych obiektów. Dla przeprowadzenia analizy funkcjonowania obiektów w ruchu, np. wewnętrznych narządów człowieka, niezbędne są *obrazy dynamiczne*. Przykładowo, praca serca i płuc jest nierozerwalnie związana z ruchem tych narządów w przestrzeni trójwymiarowej. Aby ocenić fizjologiczny stan tych organów, wymagany jest dokładny pomiar zmian wielkości, kształtu i wydajności pracy serca i płuc.

W wypadku zastosowania konwencjonalnych tomografów niekontrolowane zmiany pozycji serca i ewntualne zmiany jego kształtu między kolejnymi projekcjami powodują błędy w rekonstruowaniu i w analizie otrzymanych wyników. Ruch badanego organu podczas trwania projekcji (co najmniej kilka sekund) powoduje nieostrość rekonstruowanego obrazu (zamazanie kształtów), pojawienie się smug i (lub) niejednorodność kontrastu. W celu uniknięcia tego typu niepożądanych efektów zastosowano technikę pomiarową zwaną *bramkowanym skanowaniem*. Polega ona na zbieraniu danych dotyczących kolejnych faz pracy serca przy wykorzystaniu EKG jako wskaźnika do selekcji informacji (projekcji), odpowiadających tej samej fazie w kolejnych cyklach pracy serca. Technika ta zakłada pełną powtarzalność pracy serca w sensie położenia, wielkości skurczu itp.

Cyklom pracy serca, podzielonym na równe przedziały czasowe Δt odpowiadają nierównomierne przedziały Δx (umownie oznaczają one zmianę położenia i kształtu serca w przestrzeni). Zatem w czasie jednego pełnego obiegu układu pomiarowego istotnych jest jedynie $1/n$ wszystkich projekcji, zawierających dane pomiarowe dla ustalonej fazy serca (n – oznacza liczbę przedziałów czasowych). Dlatego do uzyskania rekonstrukcji pełnej dynamiki serca należy wykonać n obiegów układu pomiarowego (rys. 3).

Przegląd kolejnych generacji tomografów rentgenowskich

Generacja	I	II	III	IV
Rok powstania, przeznaczenie	1972 badanie głowy	1975 badanie ciała	1977 badanie ciała	1977 badanie ciała
Liczba detektorów	pojedynczy detektor	kilkanaście detektorów położonych liniowo	system 380 750 detektorów położonych wewnątrz kolistej ramy	system 424 2000 detektorów położonych wewnątrz kolistej ramy, nie sprzężony z lampą
Charakterystyka wiązki promieniowania	promieniowanie ciągłe, wiązka ołówkowa o rozwartości 1	promieniowanie ciągłe, wiązka wachlarzowa o rozwartości 5 12	promieniowanie impulsowe o czasie trwania 1 10 ms, wiązka wachlarzowa o rozwartości 40 45	promieniowanie ciągłe, wiązka wachlarzowa
Sposób zbierania danych pomiarowych	lampa rentgenowska i sprzężony z nią detektor wykonują ruch liniowy i obrotowy	lampa rentgenowska i sprzężony z nią detektor wykonują ruch liniowy i obrotowy, szybki ruch liniowy spowodowany użyciem szerszej wiązki promieniowania	lampa rentgenowska i system detektorów wykonują ruch obrotowy wewnątrz kąta 240 360 z różnymi prędkościami	lampa rentgenowska obraca się wewnątrz systemu detektorów
Czas zbierania danych	ok. 5 min	20 60 sekund	1,4 14 sekund	1 12 sekund
Charakterystyka danych pomiarowych	projekcje równoległe dla każdego kątowego położenia lampy	projekcje dla każdego kątowego położenia lampy	pojedyncza projekcja zawiera dane pomiarowe ze wszystkich detektorów	pojedyncza projekcja zawiera dane pomiarowe z detektorów objętych wiązką promieniowania



Rys. 3. Obraz EKG i odpowiadające mu ruchy serca (a); b) dane pomiarowe (zakreskowane) dla wybranej fazy pracy serca

Przełomowym wydarzeniem dla rozwoju tomografii komputerowej [9] było zbudowanie pierwszego dynamicznego przestrzennego rekonstruktora DPR (ang. *dynamic spatial reconstructor*, DSR). Różni się on od tradycyjnych systemów tomografii komputerowej dwiema głównymi cechami:

- dane pomiarowe, zbierane pod wieloma kątami, są dostarczane równocześnie dzięki zastosowaniu wielu lamp rentgenowskich, rozmieszczonych na 160-stopniowym łuku kolistej ramy, która automatycznie obraca się wokół pacjenta;
- zamiast czujników zastosowano ekran fluorescencyjny i kamery telewizyjne, które zbierają dwuwymiarowe dane pomiarowe dla każdego źródła promieni X, w ten sposób gromadząc informacje, niezbędne dla trójwymiarowej rekonstrukcji.

Pierwsza konstrukcja DPR może równocześnie badać 240 sąsiednich poprzecznych plastrów (o grubości ok. 1 mm) w tempie do 60 projekcji na sekundę. System pomiarowy rekonstruktora składa się z 14 lamp rentgenowskich i 14 kamer telewizyjnych (z 28 zaprojektowanych) rozmieszczonych na kolistej ramie co 12 stopni (kamera naprzeciwko lampy). Podczas gdy każda lampa emituje promieniowanie X przez 350 μs, na ekranie fluorescencyjnym tworzony jest obraz o wymiarach 30×30 cm^2, następnie odczytywany przez kamerę TV w ciągu 762 μs. Ponieważ istnieje 14 układów lampa-kamera, 14 telewizyjnych obrazów tworzy jedną projekcję. DPR bada więc równocześnie cylindryczną objętość o wysokości 22 cm i średnicy 22 cm w ciągu 0,011 s (dla 14 kątów zbierania danych) lub 2,244 s (dla 240 kątów). Dawka promieniowania w DPR, związana z tworzeniem jednej projekcji jest 4–5 --krotnie mniejsza od dawki naświetlania w konwencjonalnej tomografii komputerowej, niezbędnej do uzyskania jednego przekroju poprzecznego. Czas wymagany do rekonstruowania obrazów przestrzennych, będących podstawą analizy dynamiki organów zależy od liczby i rozmiaru pojedynczych przekrojów oraz liczby kątów zbierania danych. Średni czas obliczeń dla pojedynczego przekroju jest równy w przybliżeniu 2 s (z czego ok. 1,2 s dla obliczeń i 0,8 s dla operacji zapisu i odczytu danych na dysk), a więc rekonstruowanie trójwymiarowego obrazu (składającego się ze 120 przekrojów) trwa ok. 4 min.

Wstępne przetwarzanie danych pomiarowych

Przed rozpoczęciem rekonstruowania obiektu, dane z projekcji zebrane przez system detektorów muszą być wstępnie przetworzone, aby skorygować błędy wynikające z niedoskonałości elementów mechanicznych i elektronicznych układu pomiarowego.

Algorytmy rekonstruowania zakładają dostępność pełnego zbioru projekcji $g_\varphi(R)$ dla dwuwymiarowej funkcji tłumienia promieniowania $\mu(x, y)$, zgodnie z równaniem:

$$g_\varphi(R) = \int_\varphi \mu(x,y)dl$$

gdzie R jest odległością od początku układu współrzędnych prostej rzutującej 1, nachylonej pod kątem φ. Dane pomiarowe pochodzące z tomografów odbiegają jednak znacznie od założeń teoretycznych, gdyż są obciążone błędem spowodowanym efektem nieliniowości, szumami pomiarowymi oraz brakiem niektórych projekcji lub ich fragmentów.

W rentgenologicznej tomografii komputerowej każdy pomiar może być opisany następującą zależnością:

$$I = I_o \exp(-\int_\varphi \mu(x,y)dl)$$

gdzie $\mu(x,y)$ jest rekonstruowaną funkcją pochłaniania promieniowania, I_o – natężeniem źródła promieniowania, I – zmierzoną wartością natężenia promieniowania po jego przejściu przez badany obiekt, l – promieniem rzutującym, nachylonym pod kątem φ. Dwie powyższe zależności można wyrazić jeszcze inaczej wzorem:

$$g_\varphi(R) = \ln\frac{I_o}{I} = \int_\varphi \mu(x,y)dl$$

Aby otrzymać projekcję zgodną z wzorem teoretycznym, należy każdą daną uzyskaną z pomiaru poddać zlogarytmowaniu, tj. operacji nieliniowej.

Idealnym źródłem promieniowania rentgenowskiego byłoby promieniowanie monochromatyczne, w którym wszystkie fotony miałyby tę samą energię. Jednakże w tomografii rentgenowskiej źródłem promieniowania jest lampa rentgenowska generująca widmo polichromatyczne zawierające zarówno fotony o wysokiej energii zwane *twardymi*, jak i fotony niskoenergetyczne, tzw. *miękkie*. Dlatego też charakterystyka rzeczywistego pochłaniania promieniowania rentgenowskiego przechodzącego przez jednorodny obiekt, w funkcji jego grubości jest nieliniowa. Efektem zjawiska utwardzania wiązki są smugi występujące na obrazie za elementami o większym współczynniku tłumienia.

Pomiary otrzymane z projekcji są obciążone również błędami, wynikającymi z nieliniowości detektorów oraz ich skończonych wymiarów. Ponadto należy także korygować błędy geometrii systemu detektorów oraz różnicę czułości kawałów, wynikającą z rozmieszczenia detektorów w systemie (przejście od wiązki wachlarzowej, stosowanej w układzie pomiarowym do wiązki równoległej, stosowanej w algorytmach rekonstrukcji).

Po wstępnym przetworzeniu danych następuje rekonstruowanie obrazu.

Metody rekonstruowania stosowane w tomografii komputerowej można podzielić na cztery podstawowe grupy:

- metody sumacyjne,
- metody wykorzystujące transformacje fourierowskie,
- metody oparte na analitycznych rozwiązaniach równań całkowych,
- metody algebraiczne (oparte na szeregach potęgowych).

Zostały one szczegółowo omówione m. in. w artykułach [3], [6–8].

Wizualizacja zrekonstruowanego obrazu

Wizualizacja zrekonstruowanych (trójwymiarowych) obiektów obejmuje:

- detekcję pewnych cech obiektu,
- identyfikację i pomiar kształtu obiektu,
- określenie zależności przestrzennych między podobiektami.

W tomografii komputerowej wykorzystuje się dwa podstawowe sposoby wizualizacji obiektów trójwymiarowych.

Pierwszy z nich, zwany *projekcją cyfrową*, polega na przedstawieniu trójwymiarowego obiektu w postaci dwuwymiarowego rozkładu jasności obrazu. Idea tego procesu polega na sumowaniu gęstości wokseli obiektu wzdłuż pęku prostych rzutujących obiekt na płaszczyznę projekcji. Przekrój obiektu tworzony w ten sposób jest bardzo podobny do konwencjonalnych zdjęć rentgenowskich. Innym sposobem wizualizacji struktur przestrzennych jest określenie powierzchni tych struktur, a następnie wyświetlenie ich na płaszczyźnie ekranu monitora z jednoczesnym uwzględnieniem niewidocznych krawędzi oraz nałożeniem efektu przezroczystości i cieniowania. Metoda ta jest szczególnie przydatna do określenia wzajemnego położenia między wizualizowanymi strukturami. Metody wizualizacji wewnętrznych struktur trójwymiarowej sceny zostały szczegółowo opisane w literaturze [2], [5].

dokończenie na s. 25

WACŁAW ISZKOWSKI
Instytut Informatyki
Politechnika Warszawska

POLSKIE TOWARZYSTWO INFORMATYCZNE

Przyspieszanie obliczeń przez ich rozpraszanie (3)

Podsumowanie

W ostatniej części artykułu podsumowano rozważania dotyczące rozpraszania obliczeń.

STRUKTURY OBLICZENIOWE I AUTOMATY KOMÓRKOWE

Przez to nowe pojęcie – struktura obliczeniowa – rozumiem wszystkie rodzaje architektur komputerów powstałe przez powiązanie klasycznych elementów architektury von Neumanna różnymi rodzajami sieci połączeń. Elementami wiązanymi w strukturę obliczeniową mogą być:
- jednostki arytmetyczno-logiczne,
- jednostki przetwarzające z modułami pamięci,
- procesory i moduły pamięci,
- procesory z modułami pamięci,
- procesory z modułami pamięci i szynami we-wy (komputery).

Sieci połączeń mogą być:
- liniowe (szynowe, sieci lokalnej),
- bezpośrednie kratowe,
- kratowe kwadratowe,
- tasowane (Benesa, bananowa, omega, delta itp.),
- wielopoziomowe ze sterowanymi połączeniami.

Zależnie od wyboru elementu i sieci uzyskuje się różnego rodzaju klasy maszyn; w pierwszej części omówiono już maszyny SISD, w których zwielokrotnionymi elementami są jednostki arytmetyczno-logiczne (skalarne i wektorowe oraz dla wartości całkowitych i zmiennoprzecinkowych).

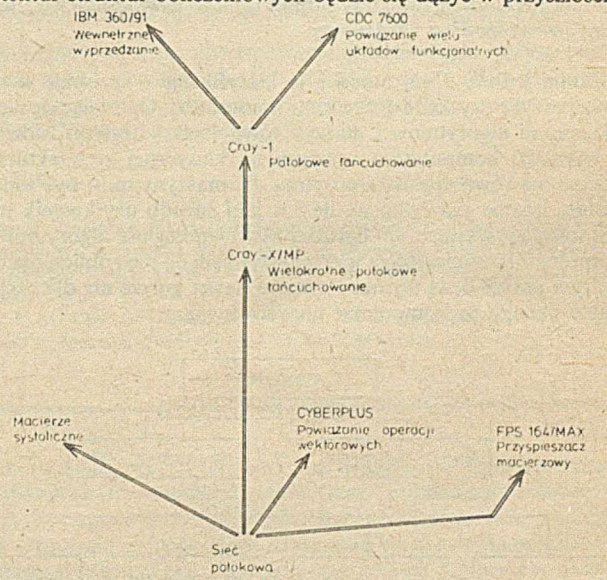
Następne są systemy klasy MIMD, czyli *n* procesorów połączonych najczęściej siecią liniową. W tej kategorii systemów *n*-procesorowych spory rozgłos zyskał transputer, czyli jednoukładowy element zawierający procesor z modułem pamięci i szyną we-wy z możliwością bezpośredniego połączenia z czterema sąsiadami (tabela 1). Obecnie trudno jest jednoznacznie ocenić zalety i wady tego układu, gdyż brakuje jeszcze odpowiednich doświadczeń. Warto jednak, zwłaszcza w kraju, ze względu na możliwość dostępu do kart ze strukturami transputerów, szerzej zainteresować się tym rozwiązaniem [1].

Tabela 1. Transputery

Transputer IMS [3] jest elementem zawierającym prosty 8-, 16- lub 32-bitowy procesor (10 MIPS) z wbudowaną lokalną wieloportową pamięcią o pojemności 2 KB z szybkością transmisji 80 Mb/s oraz z czterema dwukierunkowymi szeregowymi łączami o szybkości transmisji do 20 Mb/s. Wykorzystanie tego elementu umożliwia skonfigurowanie sieci złożonej nawet z 16 384 węzłów i mającej moc 196 GFLOPS (!). Na przykład, opracowano procesor równoległy (FPST *Series Parallel Processor*) złożony z 8 węzłów tworzących kostkę. Do tworzenia oprogramowania stosuje się język Occam (opracowany dla transputera) lub VAX/VMS Fortran V.4.2.
Dostępne są też następujące karty do IBM PC/AT wykorzystujące serię transputerów T414 (15 MHz i 20 MHz) oraz T800 (20 MHz i 25 MHz):
Monoputer2 (20 MHz T414 z 16 MB pamięci)
Monoputer2 (20 MHz T800 z 16 MB pamięci)
Monoputer2 (25 MHz T800 z 16 MB pamięci)
Biputer (20 MHz T414 + 20 MHz T800 z 2 MB pamięci)
Biputer (2 × 20 MHz T800 z 2 MB pamięci)
Quadputer (4 × 20 MHz T800 z 4 MB pamięci)
Quadputer (4 × 20 MHz T414 z 4 MB pamięci)
Interesującym przykładem transputerowej struktury obliczeniowej jest też Cogent XTM (wielokrotne T800) z rekonfigurowalną (w ciągu 200... 400 μs) siecią połączeń, umożliwiającą bezpośrednie połączenie każdego transputera z każdym innym w czasie do 40 μs.
Przykładowo, instalacja dla Sandia National Laboratory jest złożona z 1900 transputerów i ma moc Craya X-MP (1900 × 3 MFLOPS).

Pozostała jeszcze klasa maszyn SIMD, zwanych również **komputerami macierzowymi,** w której podstawową zasadą jest istnienie wielu prostych elementów przetwarzających, sterowanych tym samym strumieniem rozkazów, ale działających na całkowicie lub częściowo odrębnych strumieniach danych. Elementy te mogą się ze sobą komunikować przez pamięć wspólną lub bezpośrednio przez sieć połączeń. Problem doboru struktury sieci oraz jej złożoności, wprowadzanego przez nią opóźnienia w przekazywaniu informacji, a także odporności na uszkodzenia jest obecnie najczęściej analizowanym zagadnieniem.

Nie sposób przedstawić tutaj nawet najważniejszych rozwiązań tego rodzaju architektur. Na szczęście są one przeważnie dość dobrze opisane w dostępnej literaturze. Znacznie istotniejsze jest pytanie, jakie rodzaje architektur są aktualnie badane (tj. preferowane po analizie uzyskanych wyników badań efektywności ich działania) oraz do jakiego rodzaju architektur struktur obliczeniowych będzie się dążyć w przyszłości.



Rys. 1. Zestawienie tendencji w opracowywaniu superkomputerów

Tabela 2. Przykłady komputerów komórkowych

W Princeton University [4] zaprojektowano i zbudowano specjalizowany komputer ze specjalnie wykonanych układów VLSI. Jedna kostka zawiera 37 procesorów ułożonych w sześciokącie z 84 zewnętrznymi końcówkami do połączenia z innymi. Jeden procesor jest więc połączony 6 liniami wejściowymi i 6 wyjściowymi z 6 sąsiadami. Do przyspieszenia komunikacji między sąsiednimi procesorami wykorzystano rejestr przesuwający. Trwają też próby ze zwielokrotnieniem rejestrów tak, aby kolejne generacje cząstek można było przetwarzać potokowo. W aktualnej wersji każda kostka może przetwarzać do 20 milionów komórek na sekundę. Przygotowaniem obliczeń – dostarczaniem danych i odbieraniem wyników – zajmuje się stacja robocza Sun 3, która wprowadza ograniczenia przetwarzania do 2/3 miliona komórek na sekundę przez jedną kostkę.
Na nieco innej zasadzie do rozwiązania tego problemu wykorzystuje się Connection Machine (CM). Jest ona złożona z 65 536 procesorów, z których każdy może być połączony z 12 sąsiadami tworząc hipersześcian. Pojedynczy procesor jest prostym bitowym komputerkiem z własną pamięcią o pojemności 8 KB oraz ze specjalizowaną własną jednostką zmiennoprzecinkową. Każdy cykl działania CM składa się z fazy komunikacji z sąsiadem i fazy obliczenia. Pracę CM nadzoruje zewnętrzny komputer, który przygotowuje i przesyła program do każdego z procesorów równoległych, dostarcza dane oraz odbiera wyniki. Może on też dokonać rekonfiguracji CM na przykład jako planarnego komputera macierzowego (każdy procesor ma 4 sąsiadów). Możliwe jest też wpisanie do każdego z procesorów wieloprocesowego programu przez powołanie na przykład 200 wirtualnych procesów równoległych na jeden procesor rzeczywisty. Autor podaje, że CM może w ten sposób symulować pracę 550 mln procesorów równoległych, a więc uaktualnić stan 10^9 komórek na sekundę.

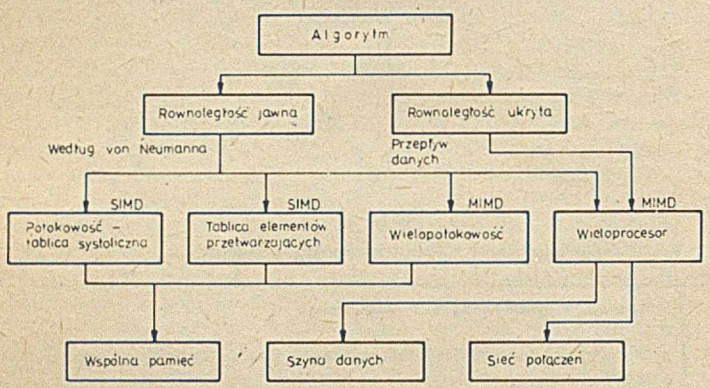
Na to ostatnie pytanie można odpowiedzieć dość prosto – wszystkie dotychczas opracowane najszybsze elementy maszyn zostaną włożone do jednego supernadkomputera, będącego wielopotokową siecią obliczania funkcji złożonych wektorowo (ang. *Multiple Networking for Fast Evaluation of Vector Compound*). Maszyna ta będzie (a może już jest?) złożona z wielu potokowych zespołów funkcjonalnych połączonych przez buforową sieć przełączającą, również potokową. Struktura połączeń w sieci ma być rekonfigurowana na dany problem obliczeniowy na podstawie jednego grafu przepływu danych. Zestawienie dotychczasowych tendencji w opracowywaniu kolejnych – handlowo uznanych superkomputerów – prowadzące do omawianego supernadkomputera pokazano na rys. 1.

Kontynuując krótki przegląd najnowszych osiągnięć w dziedzinie sprzętowego rozpraszania obliczeń, warto wspomnieć o pracach nad specjalizowanymi automatami komórkowymi przeznaczonymi do symulowania złożonych zjawisk fizycznych. Ze względu na brak miejsca nie będę ich jednak omawiał odsyłając czytelników np. do lektury czasopisma Byte [4] (p. tabela 2).

ŚWIADOMOŚĆ ROZPRASZANIA

Omawiając temat przyspieszania obliczeń przez ich rozpraszanie patrzyłem na ten problem raczej z punktu widzenia projektanta takiego systemu cyfrowego, w mniejszym stopniu zajmując się problemami, z jakimi spotyka się użytkownik takiego systemu. Rodzi się jednak pytanie, na ile musi on znać i rozumieć cechy funkcjonalne sprzętu i oprogramowania, aby móc uzyskać jak najlepsze przyspieszenie obliczeń? Rysują się tutaj trzy stopnie takiego uświadomienia – **nieświadomy, uświadamiany** i **świadomy**.

Zanim jednak bliżej omówię tę klasyfikację – spróbuję w skrócie podsumować uzyskane dotychczas wiadomości. Opierając się na jednej taksonomii algorytmów i maszyn równoległych (rys. 2) widzimy, że w wypadku komputerów opartych na klasycznej architekturze von Neumanna równoległość algorytmu dla maszyny musi być zadawana jawnie. Istotne jest tylko na ile i w jaki sposób użytkownik musi tę równoległość wyrazić. Im bardziej zaś równoległość algorytmu będzie niespójna z równoległością wykorzystywanej do jego obliczenia maszyny, tym gorsze będą wyniki – czasem nawet gorsze niż dla obliczenia realizowanego na komputerze uniwersalnym.



Rys. 2. Taksowanie algorytmów i maszyn równoległych

Przy okazji zaprezentowanego rysunku muszę wspomnieć o koncepcji realizacji obliczeń równoległych w komputerach sterowanych przepływem danych. W ich wypadku określenie równoległości obliczeń wynika w naturalny sposób z zasady działania takiego komputera. Niestety rozwiązanie to nie znalazło większego zastosowania oprócz instalacji badawczych. Inną sprawą jest konieczność zmiany sposobu programowania dla tych komputerów, co na pewno nie spotka się z entuzjazmem u wielu dotychczasowych użytkowników komputerów klasycznych.

Z drugiej jednak strony program sterowany przepływem danych może być efektywnie wykonywany w wieloprocesorach. Każdy z procesorów jest wtedy uniwersalną jednostką do wykonania prostego lub złożonego obliczenia. Rodzaj obliczenia (identyfikator operacji) oraz konieczne dla niego argumenty są pobierane z puli gotowych operacji zlokalizowanych we wspólnej pamięci. Cechą charakterystyczną procesów realizujących obliczenia jest fakt, że odwiedzając monitor w poszukiwaniu nowej operacji do wykonania – nigdy się w nim nie zawieszają. Zwielokrotnienie liczby takich monitorów – punktów dostarczających prace dla procesorów – pozwala uniknąć nasycenia programowego. Brak kosztownych czasowo operacji zawieszania i odwieszania procesów również zmniejsza straty efektywności działania systemu. Taki system zwany *„Flying Processes"* jest znacznie efektywniejszy niż realizacja klasyczna z centralnym monitorem i operacjami *delay* i *continue* [2].

Brak świadomości rozpraszania

Najprostszy dla użytkownika jest brak konieczności uświadamiania sobie istnienia jakichś wyrafinowanych metod rozpraszania obliczeń. Znając jednak dotychczasowe rozwiązania sprzętowe i programowe wiemy, że jest to jeszcze postulat nierealny. W niewielkim bowiem stopniu udało się w pełni zautomatyzować podział programu na obliczenia równoległe. Istniejące już w tej dziedzinie dokonania wymagają współudziału programisty, gdyż przecież to on najlepiej zna opracowany algorytm i może wskazać fragmenty równoległe.

Często też gwarancją znacznego przyspieszenia obliczeń jest opracowanie odpowiedniego algorytmu w zależności od stopnia i sposobu zrównoleglenia sprzętu. Poszukiwanie takich algorytmów jest obecnie bardzo popularne. Trudno jednak sobie wyobrazić, aby ten graniczący ze sztuką proces mógł być szybko zautomatyzowany bez radykalnych postępów w rozwiązywaniu problemów sztucznej inteligencji.

Rozpraszanie uświadamiane

Wydaje się, że świadomość użytkownika dotycząca cech narzędzi do rozpraszania obliczenia powinna wystarczyć. Pozwala to przy wykorzystaniu programów wspomagających przekształcić, z udziałem użytkownika, jego program na postać zrównoleglonych obliczeń. Ułatwione jest też zadanie wyjaśnienia istnienia ograniczeń w sprzęcie. Należy sądzić, że jeszcze długo ten stopień znajomości problemu rozpraszania będzie wymagany od każdego programisty i użytkownika systemów równoległych.

Pełna świadomość rozpraszania

Pełna świadomość możliwości rozpraszania sprowadza się do konieczności dokładnego poznania bardzo skomplikowanego sprzętu i oprogramowania narzędziowego. Można tutaj odnieść się do analogii z programowaniem w języku asemblera, a czasem nawet z mikroprogramowaniem maszyn matematycznych. Z drugiej jednak strony tylko w takim wypadku można mówić o rzeczywistym pełnym wykorzystaniu ulepszonego sprzętu oraz specjalnego oprogramowania. Wydaje się więc, że jak dawniej istniały w ośrodkach zespoły analityków systemu oraz programistów pomagających użytkownikom w sprawnym posługiwaniu się systemem, tak i teraz musi istnieć podobne rozwiązanie. Nawet bowiem w komercjalnie działających ośrodkach z superkomputerami, sprawność wykonywanych tam programów nie jest zbyt duża w porównaniu z możliwościami.

Kierunki rozwojowe

Podsumowaniem tych rozważań niech będzie przedstawienie przewidywanych kierunków rozwoju oprogramowania dla systemów równoległych:

1) rozszerzanie istniejących języków programowania – zapewne Fortranu 88 oraz Pascala (i C ?), umożliwiające programiście zapisanie współbieżności i synchronizacji zadań wykonywanych z wykorzystaniem potokowego przetwarzania wektorowego;

2) rozbudowanie istniejących kompilatorów o wykrywanie i przekształcanie treści programu na zrównoleglony kod wynikowy, czyli dążenie do maksymalnego automatycznego zrównoleglenia obliczeń dla danego typu maszyny równoległej;

3) wprowadzanie metajęzyków umożliwiających tworzenie wielozadaniowych zsynchronizowanych zapisów złożonych obliczeń;

4) zaprojektowanie całkowicie nowych, obiektowych i wieloprocesowych języków, ukierunkowanych na prosty zapis nowych rozwiązań równoległych algorytmów obliczeń oraz na najnowsze rozwiązania sprzętowe (na przykład, wielopotokowe wektorowe sieci obliczeniowe).

• • •

Na zakończenie spróbuję odpowiedzieć na dwa podstawowe pytania.

1. Kiedy stosować przyspieszanie?

Najprostsza odpowiedź na to pytanie brzmi: zawsze, kiedy tylko jest to możliwe. Trudno jednak zgodzić się z takim poglądem. Oczywiście, stosując w miarę posiadanych środków odpowiednie zabiegi można często nawet znacznie przyspieszyć obliczenie. Trzeba jednak pamiętać, że wymaga to od projektanta algorytmu, programisty, operatora systemu znacznie większej pracy niż zwykle. Również translator programu działa znacznie dłużej niż normalnie. Co więcej, często dopiero po kilku (oby!) próbach różnych modyfikacji algorytmu obliczeń uzyskuje się lepsze wyniki. Wykrycie poważniejszych błędów logicznych prowadzi czasem do powtórnego poszukiwania dobrego, w sensie efektywności, rozwiązania.

Tak więc w wielu wypadkach nie ma sensu „gonić" za wątpliwym przyspieszeniem obliczeń. Z drugiej strony istnieje wiele problemów z różnych dziedzin wiedzy, których ilościowe rozwiązanie jest możliwe dopiero dzięki istnieniu bardzo szybkich komputerów. I trzeba stwierdzić, że takich problemów w samej informatyce jest niewiele, a więc odpowiedzi na to pytanie powinni w zasadzie udzielić fizycy, chemicy, meteorolodzy ..., wskazując na znane im potrzeby. Z najbardziej znanych można wymienić:

- skomplikowane strukturalne obliczenia numeryczne – modelowanie pogody, przepływy płynów, analiza elementów skończonych,
- niestrukturalne obliczenia numeryczne – symulacje Monte Carlo, problemy macierzy rzadkich,
- różne problemy czasu rzeczywistego – rozpoznawanie mowy, przetwarzanie obrazów, animacja komputerowa,
- problemy przetwarzania danych – ogromne bazy danych z systemami transakcyjnymi,
- projektowanie graficzne wspomagane komputerowo,
- sztuczna inteligencja – bazy wiedzy.

2. Jaki wybrać komputer?

Prawie na pewno mogą paść tutaj pytania o ile szybciej wykona się program bez i z koprocesorem zmiennoprzecinkowym lub wykorzystującym inną metodę banalnego rozpraszania. Podobne pytania mogą być też formułowane po poznaniu różnego rodzaju architektur superkomputerów, wieloprocesorów, komputerów macierzowych i komórkowych.

Oczywiście takie badania dla typowych mieszanek programów istnieją, a uzyskane wyniki można znaleźć bądź w reklamach firm, bądź w poważnych analizach porównawczych. Niestety, podane tam czasy wykonania mieszanek programów niwelacyjnych są dla zwykłego użytkownika mało przydatne. Pomijając bowiem wykonywanie typowych już obliczeń z użyciem programów bibliotecznych, w pozostałych wypadkach trudno jest wyrokować, który z dostępnych komputerów będzie najlepszy dla danego typu problemu. Możliwe jest jedynie pewne oszacowanie na podstawie analizy struktury algorytmu i sposobu obliczeń najczęściej wykonywanych w nim operacji. Główny wysiłek musi zostać skierowany na zaprojektowanie „na miarę" dla danego komputera odpowiedniego programu. Zdajemy już sobie sprawę, że nie jest to zadanie banalne nawet korzystając ze specjalnych translatorów i narzędzi wspomagających. Co więcej, programy te są oparte na algorytmach mało zrozumiałych dla informatyków, a z kolei obecny stan wiedzy o efektywnym wykorzystaniu superkomputera jest jeszcze dziedziną znaną tylko niewielu.

Znacznie prostsza sytuacja występuje u nas w kraju. Powinniśmy sobie jednak uświadomić następujące fakty.

- W Polsce brak jest w ogóle jakichkolwiek doświadczeń w użytkowaniu tego rodzaju maszyn, a więc też nie ma możliwości nauczenia, przynajmniej podstaw, następnych pokoleń.
- Boom mikrokomputerowy oraz brak możliwości zakupu przynajmniej VAX-a powoduje wytworzenie się przekonania, że tysiącami PS/2 zastąpi się jednego Craya.
- Możliwe jest jednak zakupienie lub zrealizowanie siłami krajowymi prostych systemów wieloprocesorowych opartych na standardowych układach mikroprocesorowych lub zintegrowanych programowo sieciach mikrokomputerowych – żałować należy, że podejmowane już w poprzednich latach próby zakończyły się typowym fiaskiem.
- Konieczne jest stałe propagowanie problematyki przyspieszania obliczeń, nawet z wykorzystaniem prostych rozwiązań, i zdobywanie wiedzy o najnowszych tendencjach w tej dziedzinie, choćbyśmy mieli czekać na Craya w kraju nawet kilka... lat.

LITERATURA

[1] Caban D.: Transputer – narzędzie przetwarzania równoległego. Informatyka, nr 6, 1988
[2] Iszkowski W.: Performance Evaluation for a Dependency Graph Computations on a MultiProcessor System. Research Reports, No. 56, Instytut Informatyki, Politechnika Warszawska, 1983
[3] Stein R. M.: T800 and Counting. Byte, Vol. 13, No. 12, November 1988
[4] Wayner P.: Modeling Chaos. Byte, Vol. 13, No. 5, May 1988.

Sieci komputerowe

Co prawda, istnieją już mikrokomputery przystosowane do łączenia w sieci i odpowiednie systemy operacyjne, jednakże ich pełne oprogramowanie i wykorzystanie nie następuje zbyt szybko. Oszacowano, że do końca 1988 roku tylko 20% użytkowników mikrokomputerów w przedsiębiorstwach przeszło na system operacyjny OS/2. Większość pozostałych pozostaje przy MS-DOS, aczkolwiek tu też następują zmiany. Dominują maszyny z mikroprocesorem Intel 80286, a użytkownicy systemów opartych na 80386 korzystają z oprogramowania Windows/386, opracowanego przez firmę Microsoft Corp. Część z nich przechodzi na OS/2 unikając trudniejszych prac do czasu opracowania programu zarządzającego Presentation Manager, co miało nastąpić w końcu 1988 roku. Wówczas również coraz więcej użytkowników systemów z mikroprocesorem 80286 będzie pracować w systemie OS/2. Na razie konieczność przystosowania się do różnych systemów operacyjnych stwarza zamieszanie w oprogramowaniu.

Podobnie dzieje się w zakresie sieci lokalnych, gdzie producenci oczekują na program LAN Manager firmy Microsoft, który pozwoli na stosowanie mikrokomputerów opartych na 80386 jako usługodawców sieciowych (ang. *servers*). Wydaje się, że usługi, jakich oczekują użytkownicy sieci lokalnych w zakresie komunikowania się w całym przedsiębiorstwie, a nie tylko w poszczególnych jego wydziałach, nie były w pełni oferowane przez sprzedawców oprogramowania dla PC w 1988 roku. Na obecnym etapie (początek 1989 roku) występują trudności z ograniczoną funkcjonalnością bram (ang. *gateway*) i niezbyt sprawnymi mechanizmami przesyłania plików. Problemy te planuje się rozwiązać przez systemy katalogów globalnych, które pozwolą na automatyczne wprowadzanie wszystkich zmian dokonanych przez użytkowników. Wówczas nadawca komunikatu nie będzie musiał znać usytuowania odbiorcy w systemie ani właściwego formatu danych, które chce przesłać w ramach systemu. Obecnie podobne możliwości ma tylko system operacyjny Vines, firmy Banyan Systems Inc., przy czym oprogramowanie tej firmy działa również na VAX-ach pracujących pod nadzorem systemu VMS. Podobne możliwości zapewniają również firmy Novell Inc. i 3 Com Corp., ale bez pełnych usług katalogu globalnego.

Ceny ogłoszeń

Od 1 stycznia 1989 r. obowiązują następujące ceny materiałów reklamowych publikowanych na łamach INFORMATYKI:

Ogłoszenia czarno-białe, artykuły reklamowe i informacje naukowo-techniczne (biuletyny) zależnie od objętości: cała strona – 70 tys., 3/4 s. – 60 tys., 2/3 s. – 55 tys., 1/2 s. – 50 tys., 1/3 s. – 45 tys., 1/4 s. – 40 tys., 1/8 tys., 30 tys., poniżej 1/8 s. – 200 zł za cm².

Dodatki do ceny podstawowej – za każdy dodatkowy kolor + 30%, za każdy specjalny kolor (nie wynikający z podstawowych kolorów) + 30%, za pełny kolor (grafika wielobarwna, zdjęcia kolorowe) + 120%, za zamieszczenie ogłoszenia na I lub IV stronie okładki + 100%, za zamieszczenie ogłoszenia na II i III stronie okładki + 50%.

Zniżki dotyczą ogłoszeń – całkowitych powtórzeń: – za ogłoszenia 3-5-krotne – 10%, – za ogłoszenia 6-10-krotne – 20%, za ogłoszenia 11-krotne i powyżej 30%, – za artykuły i wkładki reklamowe – 60%. W innych uzasadnionych wypadkach dopuszcza się stosowanie rabatów specjalnych.

Ceny nadbitek reklamowych: wkładka 2 s. A4 – nakład do 500 egz. – 30 tys. zł, za każde następne 500 ezg. – 25 tys. zł; wkładka 4 s. A4 – nakład 500 egz. – 60 tys. zł, za każde następne 500 egz. – 50 tys. zł.

Ceny usług nie wymienionych w niniejszym cenniku będą ustalane każdorazowo na podstawie kalkulacji, jako ceny umowne. W wypadku rezygnacji Zleceniodawcy z wykonania zamówienia przed przekazaniem materiałów do druku – ponosi on koszty w wysokości 25% wartości zlecenia. W wypadku rezygnacji – gdy materiały są już w druku – Zleceniodawca ponosi pełne koszty ogłoszenia.

Niniejszy cennik dotyczy wyłącznie ogłoszeń firm krajowych i obowiązuje od 1 stycznia 1989 r. Ogłoszenia przyjęte przed tym terminem będą rozliczane według dotychczas obowiązującego cennika.

Ogłoszenia przyjmowane są przez:
Dział Ogłoszeń i Reklamy WCiKT NOT SIGMA,
ul. Świętojerska 5/7, 00-236 Warszawa
adres do korespondencji: skrytka pocztowa 1004, 00-960 Warszawa
telefony: 31-93-65 lub 31-22-21 w. 196 i 291

NEWCOMP SA

ul. Piechoty Łanowej 58, 02–951 Warszawa
tel. 42 25 58

PROPONUJEMY:

- IBM seria 400
- VAX, MICROVAX
- SUN
- PC/AT/XT 386
- PS 2
- PC MOS 386, UNIX, oprogramowanie aplikacyjne
- drukarki mozaikowe, laserowe, elektrostatyczne
- pamięci masowe o dużych pojemnościach
- systemy automatyzacji produkcji
- systemy ekspertowe, bazy wiedzy
- serwis urządzeń informatycznych
- wprowadzanie i konwersja danych

WPROWADZAMY jako wyłączny dystrybutor
pierwszy w Polsce, rewelacyjny
SYSTEM MAGAZYNOWANIA DANYCH

VIDEOTRAX

- wykorzystanie standardowego zapisu wizyjnego do magazynowania danych
- zdumiewająca dokładność zapisu – prawdopodobieństwo powstania błędu jest niższe jak w twardym dysku
- tańszy w eksploatacji:
  1 kaseta streamera = 150 tys. zł
  1 kaseta videotraxu VHS = 12 tys. zł
- 1 kaseta 120 min. VHS = 160 MB danych

EO/1096/88

ZDZISŁAW SZCZERBIŃSKI
Zakład Systemów Automatyki Kompleksowej
Polska Akademia Nauk
Gliwice

Nowa generacja superkomputerów Cray (2)

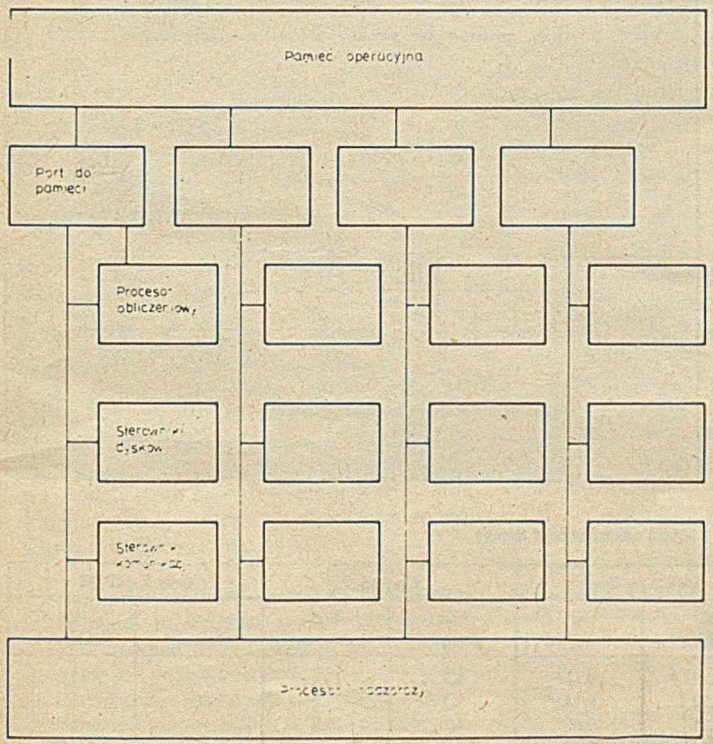
Cray-2

W 1981 r. Seymour Cray opuścił kierownicze stanowisko w założonej przez siebie w 1972 r. firmie Cray Research i pracuje w niej odtąd jako niezależny projektant „na kontrakcie". Mogąc poświęcić więcej czasu na realizację własnych pomysłów, zaprojektował i w ciągu kilku następnych lat doprowadził do produkcji modelu Cray-2. Jednoprocesorowe prototypy tej maszyny pojawiły się na przełomie lat 1984 i 1985. Pierwszy czteroprocesorowy Cray-2 zaprezentowano w połowie 1985 r. Charakterystyczne cechy tego superkomputera to bardzo krótki czas cyklu zegara, bardzo duża pamięć operacyjna oraz niekonwencjonalny sposób chłodzenia – zanurzenie pakietów systemu w cieczy fluorowęglowej.

ARCHITEKTURA

Na rys. 1 przedstawiono ogólny schemat blokowy systemu Cray-2. Sekcję obliczeniową stanowią cztery procesory obliczeniowe, pracujące w tle (ang. *background processor*), każdy z nich mocą obliczeniową przewyższa komputer Cray-1. Mają one dostęp do wspólnej pamięci przez oddzielne porty. Procesor nadrzędny nadzoruje pracę całego systemu i koordynuje przepływ danych między pamięcią, procesorami obliczeniowymi i urządzeniami zewnętrznymi, za pośrednictwem czterech szybkich kanałów.

Struktura procesora obliczeniowego (wektorowego) jest podobna do istniejącej w modelu Cray X-MP. Różnice są następujące:
- tylko jeden port do pamięci (jak w Cray-1),
- tylko dziewięć jednostek funkcjonalnych (potokowych), za to w większym stopniu wielofunkcyjnych,



Rys. 1. Cray-2 – schemat ogólny

- lokalna pamięć 16 K słów 64-bitowych,
- brak bufora skalarnego i bufora adresów (B, T),
- brak sekcji we-wy,
- 64-bitowy programowany zegar czasu rzeczywistego (zamiast 32-bitowego).

Czas cyklu zegara wynosi tylko 4,1 ns (najkrótszy na świecie). Dodano nowe rozkazy: przybliżenie pierwiastka kwadratowego, konwersja liczby całkowitej na zmiennoprzecinkową i odwrotnie, zapis i odczyt do lub z pamięci lokalnej.

Wspólna pamięć operacyjna obejmuje 256 M słów (2 GB). Jest to największa pamięć operacyjna na świecie (nie uwzględnia się w tej klasyfikacji pamięci wirtualnych, poza systemem CDC Cyber 205 w superkomputerach nie stosowanych). Składa się na nią 128 modułów, z przeplotem adresowania, każdy moduł zawiera 2 M słowa. Podobnie jak we wcześniejszych modelach Cray, 64-bitowe słowo jest uzupełnione 8 bitami korekcji (SECDED). Pamięć jest dzielona między procesory obliczeniowe, procesor nadzorczy i sterowniki urządzeń peryferyjnych przez cztery wspólne porty. Przeplot adresów jest możliwy dzięki niezależnym drogom przesyłania danych między każdym modułem pamięci i każdym z portów. Szybkość przesłań z i do pamięci wynosi 64 Gb/s.

Procesor nadzorczy, zarządzający całym systemem, jest dołączony do czterech synchronicznych kanałów komunikacyjnych o szybkości 4 Gb/s każdy. Każdy z kanałów łączy procesor nadzorczy z jednym procesorem obliczeniowym, jedną grupą sterowników we-wy i jednym portem pamięci. Do systemu można dołączyć do 40 urządzeń peryferyjnych (m. in. 36 jednostek dyskowych).

TECHNOLOGIA I OPROGRAMOWANIE

System Cray-2 skonstruowano z 16-bramkowych układów scalonych w technologii ECL (obwody logiczne) i pamięci MOS. Aby zminimalizować długość połączeń (podstawowy problem konstruktorów superkomputerów), maksymalnie upakowano elementy w 320 modułach. Każdy moduł zawiera ok. 750 układów scalonych. Łączna liczba układów w systemie wynosi ok. 240 000, z czego ok. 75 000 przypada na pamięć. Moduły są trójwymiarowe, o grubości 1 cala (osiem złożonych pakietów; bezpośrednie połączenia między elementami sąsiadujących pakietów – stąd trzeci wymiar połączeń). Każdy moduł waży prawie 1 kg i zużywa 300 do 500 W energii elektrycznej (całkowite zużycie w systemie – 195 kW).

Zewnętrznie, jednostka centralna to 14 pionowych kolumn, o wysokości 1,14 m, połączonych w łuk 300° (niedomknięty walec). Górna część każdej kolumny zawiera 24 moduły, dolna – zasilacze. Średnica podstawy jednostki centralnej wynosi 1,35 m, co sprawia, że zajmuje ona tylko 1,5 m² powierzchni podłogi; waży jednak 2,5 T.

Niezwykle gęste upakowanie elementów wymaga specjalnego chłodzenia. Walec jest wypełniony cieczą chłodzącą, w której moduły są bezpośrednio zanurzone (ang. *liquid immersion cooling*). Ciecz ta jest związkiem fluorowęglowym, nietoksycznym, niepalnym i, oczywiście, dielektrycznym. Opływa ona moduły z prędkością 1 cala/s, będąc w bezpośredniej styczności ze wszystkimi powierzchniami pakietów oraz układami zasilania.

System operacyjny maszyny Cray-2 – UNICOS (możliwa jest praca w tym systemie również dla Cray X-MP) jest opracowaną przez firmę Cray Research wersją Unixa – System V. Jądro zostało rozwinięte dla

potrzeb operacji we-wy, efektywnego użytkowania dużych zbiorów danych, wieloprocesorowości i wielozadaniowości. W ramach swoich zadań, użytkownicy mogą tworzyć procesy współbieżne. UNICOS ma standardową powłokę (ang. *shell*) procesora poleceń – użytkownicy mogą też tworzyć własne powłoki. Pozostałe oprogramowanie systemowe to, podobnie jak w systemie Cray X-MP, kompilatory języków Fortran, Pascal, C i asembler CAL (wersja 2.0).

MOC OBLICZENIOWA MASZYN CRAY

Tradycyjne miary mocy obliczeniowej dużych maszyn dotyczą szybkości przetwarzania. Określa się ją w milionach rozkazów wykonanych w ciągu sekundy MIPS (ang. *Million Instructions Per Second*) lub milionach operacji zmiennoprzecinkowych na sekundę MFLOPS (ang. *Million Floating-Point Operations Per Second*). W świecie superkomputerów, maszyn projektowanych z myślą o zastosowaniach inżynierskich i naukowych, w których dominują działania na liczbach rzeczywistych, przyjęto drugi z wyżej wymienionych parametrów za bardziej miarodajny.

Maksymalna szybkość maszyny Cray-1 wynosiła 160 MFLOPS. Model X-MP jest komputerem wieloprocesorowym; dla jednego procesora maksymalna szybkość wynosi 210 MFLOPS, tak więc w konfiguracji dwuprocesorowej Cray X-MP/2 osiąga się szybkość 420 MFLOPS, a w czteroprocesorowym Cray X-MP/4 – 840 MFLOPS. Cray-2 osiąga maksymalnie 500 MFLOPS w jednym procesorze, a więc 2 GFLOPS przy równoczesnej pracy wszystkich czterech procesorów obliczeniowych. Są to jednak szybkości teoretyczne. W praktyce, obliczenia są wykonywane znacznie wolniej. Praktyczna szybkość zależy bowiem od wielu czynników: długości wektorów, liczby operacji skalarnych, liczby dostępów do pamięci i in. Od strony programowej, najistotniejsze znaczenie ma złożoność obliczeniowa algorytmu oraz sposób jego zakodowania (decydują tu umiejętności programisty).

W chwili obecnej brak jeszcze reprezentatywnych danych na temat praktycznej szybkości działania systemu Cray-2, są natomiast dość liczne wyniki programów wzorcowych dla modelu Cray X-MP, realizowanych jednak na jednym tylko procesorze. W tabeli 1 przedstawiono rzeczywistą szybkość systemu X-MP dla różnych działań na wektorach, w funkcji długości wektorów (iteracji pętli) [5]. Warto zwrócić uwagę na ostatnie dwie operacje: mała szybkość jest wynikiem pośredniego adresowania argumentów, uniemożliwiającego wektoryzację pętli.

Tabela 1. Szybkość pracy Cray X-MP, w MFLOPS, dla wybranych działań na wektorach

Działanie	Długość wektora				
	10	50	100	200	1000
A(I) = B(I) + S	14,4	57,6	63,9	69,5	76,6
A(I) = B(I) * C(I)	14,0	53,5	58,6	61,6	68,7
A(I) = B(I) * C(I) + D(I) * E(I)	33,5	97,7	102,5	108,6	115,2
S = S + A(I) * B(I)	5,2	21,7	36,9	58,6	117,4
A(I) = B(J(I)) + S	2,3	2,5	2,5	2,5	2,5
A(J(I)) = B(I) * C(I)	3,1	3,5	3,5	3,5	3,5

Interesujące jest również zestawienie szybkości superkomputerów Cray z najpopularniejszymi dużymi systemami firmy IBM, nie będącymi jednak superkomputerami, oraz superminikomputerem VAX-11. Porównanie takie [1] przeprowadzono w Los Alamos National Laboratory w USA dla opracowanych tam programów wzorcowych, napisanych w języku Fortran i realizujących typowe algorytmy inżynierskie (rozwiązanie układu równań liniowych, szybka transformata Fouriera, metoda Monte Carlo itp.). W maszynach IBM wykorzystano optymalizacyjny kompilator H/OPT3.

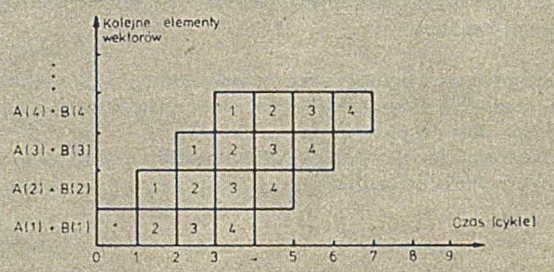
dokończenie na s. 8

Przetwarzanie potokowe

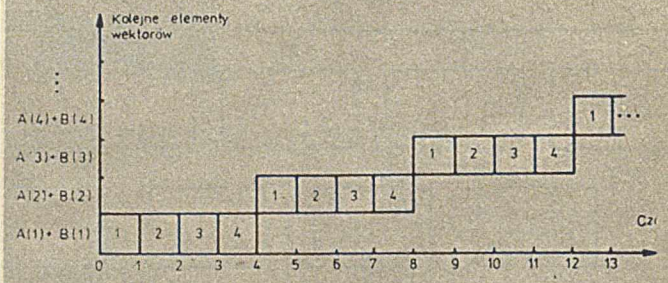
Koncepcję przetwarzania potokowego najłatwiej wyjaśnić, odwołując się do analogii z fabryczną linią montażową. Jeśli podzielimy proces produkcyjny na następujące po sobie kolejno fazy i potrafimy na określonym stanowisku pracy wykonać daną fazę produkcyjną niezależnie od innych, to zakładając identyczny czas trwania każdej fazy, można pracować równocześnie nad tyloma produktami, ile jest faz. W odstępie czasu przypadającym na fazę (cyklu), produkt ulega obróbce w stanowisku i, w następnym cyklu będzie on przetwarzany w stanowisku $i+1$, a w stanowisku i znajdzie się następny produkt, przysłany ze stanowiska $i-1$. W ten sposób gotowy produkt otrzymujemy co jeden cykl. W komputerowych jednostkach przetwarzania potokowego (jednostki funkcjonalne w maszynach Cray) stanowiskami pracy (etapami) są układy kombinacyjne, wykonujące operacje arytmetyczno-logiczne na przepływającym strumieniu danych. Układy te są od siebie oddzielone szybkimi rejestrami, przechowującymi wyniki pośrednie, generowane w poszczególnych etapach. Przepływem danych między etapami steruje wspólny zegar, odblokowujący wyjścia wszystkich rejestrów w tym samym momencie.

Klasycznym przykładem przetwarzania potokowego jest sumowanie dwóch wektorów liczb zmiennoprzecinkowych. Operacja dodawania dwóch liczb zmiennoprzecinkowych dzieli się na następujące cztery etapy:
1) porównanie cech,
2) przesunięcie jednej z mantys o różnicę cech,
3) dodanie mantys,
4) normalizacja wyniku.

Przebieg czasowy dodawania wektorów przez potokowy sumator zmiennoprzecinkowy przedstawiono na rys. 2, a przez sumator konwencjonalny na rys. 3.



Rys. 2. Dodawanie wektorów przez potokowy sumator zmiennoprzecinkowy



Rys. 3. Dodawanie wektorów przez sumator konwencjonalny

Tabela 2. Szybkość superkomputerów Cray w porównaniu z systemami IBM i VAX (w MFLOPS – kreski oznaczają brak testów)

Numer testu	IBM 370/195		IBM 3033		VAX 11/780		Cray-15		Cray X-MP/24	
	32 bity	64 bity	32 bity	64 bity	32 bity	64 bity	Kod skalarny	Kod wektorowy	Kod skalarny	Kod wektorowy
1	2,10	2,00	1,50	1,40	0,17	0,11	6,3	11,1	8,3	15,6
2	2,64	–	–	–	0,22	0,14	7,1	11,3	8,7	13,5
3	2,60	2,60	1,50	1,30	0,11	0,06	3,3	18,7	4,2	31,9
4	–	–	–	–	0,11	0,07	3,0	12,9	4,0	20,0
5	–	–	–	–	0,20	–	6,2	14,2	6,7	18,8

Stanowisko Polskiego Towarzystwa Informatycznego w sprawie ochrony prawnej programów komputerowych

1. W środowisku Polskiego Towarzystwa Informatycznego podnoszona jest od dłuższego czasu teza o konieczności ustanowienia jednoznacznej ochrony prawnej programów komputerowych. Przemawiają za tym – zgodnie z opinią środowiska – następujące argumenty:
– ustanowienie takiej ochrony jest jednym z koniecznych warunków dla rozwoju krajowego przemysłu oprogramowania, bez istnienia którego nie może być mowy o postępie zastosowań informatyki w Polsce;
– obserwowane w skali masowej zjawisko piractwa w dziedzinie oprogramowania powoduje wiele szkód w gospodarce narodowej, poprzez fakt częstego używania oprogramowania niekompletnego i niewłaściwie udokumentowanego lub uszkodzonego w wyniku łamania zabezpieczeń;
– kopiowanie na szeroką skalę oprogramowania zagranicznego bez zgody i woli producentów stanowi łamanie zasad przyjętych w obrocie w ich krajach i jest poważną barierą dla rozwoju prawidłowych stosunków handlowych z tymi producentami;
– sytuacja, w której można za darmo otrzymać kopie programów, stwarza społeczne przekonanie, że oprogramowanie nic nie kosztuje lub kosztuje niewiele. To przekonanie jest błędne i może mieć poważne skutki dla podejmowanych decyzji gospodarczych.

2. Konieczność ustanowienia ochrony prawnej oprogramowania jest ściśle związana z prowadzonymi analizami oraz wypowiedziami przedstawicieli nauki prawa, z których wynika, że obowiązujący w Polsce prawny system ochrony własności intelektualnej (prawo autorskie, prawo wynalazcze) jest w niedostatecznym stopniu przystosowany do tego, aby mógł być podstawą jednoznacznej kwalifikacji programów komputerowych, jako przedmiotu ochrony. Fakt ten dostrzeżono w krajach wytwarzających oprogramowanie komputerowe na skalę przemysłową, dokonując stosownej nowelizacji obowiązującego prawa autorskiego. Sugestie w tym zakresie wynikają ponadto z prac Światowej Organizacji Własności Intelektualnej, której członkiem jest Polska.

3. Nie przesądzając ostatecznych rozważań legislacyjnych Polskie Towarzystwo Informatyczne podjęło prace nad zdefiniowaniem podstawowych elementów podmiotowego prawa bezwzględnego (prawo własności programu), mającego za przedmiot programy komputerowe. Podstawowe elementy definicji proponowanej przez Towarzystwo są załączone do tekstu niniejszego stanowiska.

4. Polskie Towarzystwo Informatyczne prezentuje pogląd, że ochrona prawna programów komputerowych, będąc warunkiem koniecznym, nie jest jedynym wystarczającym warunkiem rozwoju przemysłu oprogramowania. Warunkami tego rozwoju są ponadto: właściwa polityka kredytowa i podatkowa, umożliwiająca odpowiednie inwestycje oraz właściwe dla przedmiotu produkcji zasady rozliczania jej kosztów.

Wprowadzane w kraju nowe zasady prowadzenia działalności gospodarczej oraz nowe zasady polityki finansowej powinny uwzględniać specyfikę produkcji oprogramowania, jej technologię oraz charakter powstających produktów. Możliwość uwzględnienia tej specyfiki jest ściśle związana z uregulowaniami dotyczącymi stosunków własnościowych, tj. prawa własności programów.

5. Wysokość opłat licencyjnych na rzecz producentów z II obszaru płatniczego, wymaganych od chwili wprowadzenia ochrony prawnej oprogramowania, przytaczana jako argument przeciwko jej ustanowieniu, jest – zdaniem Towarzystwa – w znacznym stopniu wyolbrzymiona. Jest ona oceniana przy założeniu konieczności wykorzystania wszystkich produktów (programowych oraz rozmaitych ich wersji), które pojawiły się na rynku krajowym w wyniku pirackiego kopiowania. Potrzeby zastosowań informatyki wymagają znacznie mniejszej liczby tych produktów, sprowadzanych jedynie w określonych wersjach.

Przy takim założeniu wysokość tych opłat należy szacować na znacznie niższym poziomie. Konieczność wnoszenia opłat stanowić może natomiast istotny czynnik dla prowadzenia ekonomicznie uzasadnionej selekcji oprogramowania importowanego i doprowadzić do praktyki dokonywania transakcji z mniejszą liczbą producentów, co może mieć korzystny wpływ w zakresie typizacji oprogramowania produkowanego w kraju.

PROPOZYCJA DEFINICJI PODSTAWOWYCH ELEMENTÓW PRAWA WŁASNOŚCI PROGRAMU

1. Przedmiotem prawa własności programu jest każdy program komputerowy ustalony w postaci:
a) kodu programu w wersji do bezpośredniego wykonania przez komputer,
b) tekstu programu w języku symbolicznym lub innej postaci, umożliwiającej automatyczne wygenerowanie kodu programu w wersji do bezpośredniego wykonania przez komputer.
1.1. Przedmiotem prawa własności programu jest również każdy szczegółowy opis programu, tj. taki opis, który umożliwia proste odtworzenie którejkolwiek z postaci, wymienionych w p. 1.

2. Podmiotem prawa własności programu jest osoba, która poniosła nakłady na opracowanie programu. W szczególności, jeżeli program opracowany został w wykonaniu umowy o pracę lub umowy o dzieło, prawo własności programu służy pracodawcy lub zamawiającemu.
2.1. Jeżeli kilka osób poniosło nakłady związane z opracowaniem programu, dla określenia prawa każdej z osób stosuje się odpowiednio przepisy prawa cywilnego dotyczące współwłasności.
2.2. Domniemywa się, że prawo własności programu służy osobie, której nazwa uwidoczniona jest w szczegółowym opisie programu lub która uwidoczniona zostaje w zewnętrznych przejawach działania programu (np. na ekranie, wydruku lub rysunku).
2.3. Bezpośrednim twórcom programu służy prawo ochrony dóbr osobistych, zgodnie z zasadami prawa cywilnego. Nie obejmuje ono jednak prawa do decydowania o sposobie wykorzystania programu przez właściciela prawa do tego programu.
3. Prawo własności programu obejmuje prawo do:
a) wyłącznego rozporządzania programem,
b) pobierania przychodów z tytułu eksploatacji programu.
3.1. Prawo własności programu może być w całości lub części zbyte innej osobie.

4. Poszczególne uprawnienia w zakresie prawa własności programu mogą być udzielone przez właściciela prawa do programu innej osobie w drodze umowy licencyjnej.

5. Korzystanie z cudzego programu może odbywać się wyłącznie na podstawie umowy zawartej z właścicielem prawa do tego programu lub osobą uprawnioną wg zasad określonych w p. 4.

6. Naruszeniami prawa własności programu są ponadto:
a) kopiowanie programu,
b) odtwarzanie tekstu programu w języku symbolicznym lub opisu programu na podstawie kodu programu w wersji do bezpośredniego wykonania,
c) wykorzystywanie istotnych rozwiązań zawartych w programie w opracowaniu innego programu, bez pisemnej zgody właściciela prawa do tego programu lub osoby uprawnionej wg zasad określonych w p. 4.

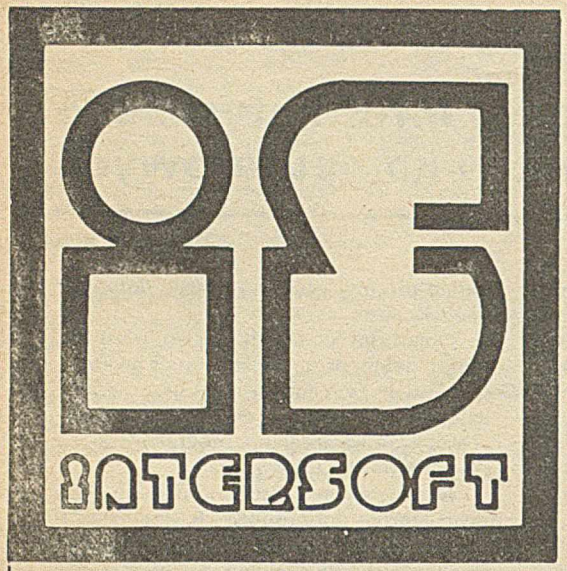
7. Opublikowanie przez właściciela prawa do programu szczegółowego opisu tego programu w książce lub czasopiśmie uważane jest za wyrażenie zgody na swobodne wykorzystywanie programu oraz zawartych w nim rozwiązań.

8. Usuwanie nazw osoby lub osób, którym przysługuje prawo własności programu oraz stwarzanie w jakikolwiek sposób przekonania, że prawo to przysługuje innej osobie lub osobom są ścigane karnie.

9. Właściciel prawa do programu lub osoba upoważniona wg zasad określonych w p. 4 może żądać od każdego, kto narusza zasady określone w p. 5 i 6, zaniechania tych naruszeń oraz wydania korzyści materialnych.

15 grudnia 1988 r.

Oprac. JACEK IRLIK



INTERSOFT

00-443 Warszawa, ul. Górnośląska 9/11
tel. 21-56-08, 28-67-94, teleks 817245

Szanowni Państwo!

Polecamy duży wybór dokumentacji dot. komputerów IBM oraz oprogramowania, m.in.:

Przewodnik programisty IBM	60 000 zł	Turbo Power Tools Plus (procedury do TP 4/5)	55 000 zł
Wprowadzenie do komputerów IBM	18 000 zł	Grafika TP v.4.0 i TC v.1.5	55 000 zł
Sidekick	23 000 zł	Clipper 86, kompil. do dBase III+	55 000 zł
Wstęp do grafiki (Basic, Turbo, Graphics)	45 000 zł	Clipper 87, kompil. do dBase III+	75 000 zł
Autocad v.2.17	50 000 zł	dBase III Poradnik encyklopedyczny	50 000 zł
Lotus 1–2–3 v.2.0	65 000 zł	dBase III+	50 000 zł
Framework IIp	70 000 zł	dBase III+, praca w sieci	20 000 zł
System operacyjny DOS 3.2	50 000 zł	Fox Base+	70 000 zł
System operacyjny DOS 3.3	85 000 zł	Programowanie w assemblerze	55 000 zł
Podręcznik programowania w GW-Basic	50 000 zł	Eureka	45 000 zł
Turbo Basic	70 000 zł	Poly-Windows	25 000 zł
Turbo „C" v.1.0	70 000 zł	Instrukcja obsługi PC 1512	35 000 zł
Turbo „C" v.1.5 t. 1 i 2	68 000 zł	Wordstar 2000	25 000 zł
Aztec „C"	60 000 zł	Modula 2 Logitech	48 000 zł
Zastosowanie języka C dla zaawansowanych	70 000 zł	Informix	90 000 zł
Programowanie w Turbo Pascal v.3.0	48 000 zł	OS-2, opis systemu	75 000 zł
Turbo Graphics (do TP v.3.0)	43 000 zł	Or-Cad	95 000 zł
Turbo Database Toolbox (do TP v.3.0)	24 000 zł	PC Tools De Luxe	15 000 zł
Turbo Pascal v.4.0	65 000 zł	Multi-Link v.4.0 (Lan Link 4.0)	50 000 zł
Turbo Pascal v.5.0	90 000 zł	Novell, podr. użytkownika i instalatora	150 000 zł

Do większości pozycji dołączamy dyskietki z przykładami (koszt nośnika nie jest ujęty w wyżej wymienionych cenach). Przy płatności czekiem lub gotówką (bądź przy sprzedaży za zaliczeniem pocztowym) udzielamy 10% zniżki.

Prowadzimy sprzedaż oraz skup wszelkiego sprzętu komputerowego oraz audio-video.

Do sprzedawanego przez nas sprzętu dodajemy bezpłatnie pakiety dokumentacji i oprogramowania o wartości zależnej od wielkości sprzedaży.

ZAPRASZAMY !

EO / 232 / 89

Wykonywanie programów DOS-a w systemie Unix

Informacja, że zestawy pracujące z Unixem na mikroprocesorze Intel 80386 realizują programy usługowe systemu DOS, może się wydawać co najmniej dziwna. To tak jakby Chevrolet lub Chrysler oferowały japońskie auta. Czy możliwe, by Lotus 1-2-3, dBase III lub WordPerfect mogły współistnieć w wielozadaniowym, wieloużytkowym środowisku obok takich programów Unixa, jak awk, grep, make czy vi? Jednak mogą, lecz aby zrozumieć jak, trzeba trochę bliżej poznać Unixa.

Czym jest Unix

Używając przenośni można Unixa nazwać mistrzem Zen informatyki. Realizuje on bowiem filozofię „małe jest piękne" i uczy, jak tworzyć swoje własne narzędzia posługując się standardowymi programami. Jednak wiele z tego, co Unix może, pozostaje dla użytkownika niewidoczne.

Jako środowisko wieloużytkowe Unix pozwala wielu osobom jednocześnie produktywnie pracować na tym samym komputerze, współużytkując dyski, drukarki i modemy. Jako środowisko wielozadaniowe Unix zarządza wieloma zadaniami lub procesami, przydzielając każdemu zadaniu odcinek czasu procesora centralnego. Gdy odcinek ten skończy się lub gdy samo zadanie stwierdzi, że musi czekać na inne zdarzenie, to system operacyjny zawiesza zadanie bieżące, podejmując w tym czasie inne. Zadania nie wykonują się od początku do końca przed rozpoczęciem następnego, lecz ich wykonanie jest przeplatane w czasie. Na zewnątrz powstaje wrażenie, że wszystkie zadania (zwane w Unixie procesami) są wykonywane równocześnie.

Wielozadaniowość wiąże się ściśle z pamięcią wirtualną. Programy, zamiast używać adresów fizycznych, pracują w praktycznie nieograniczonej przestrzeni pamięci wirtualnej. W ten sposób znacznie więcej programów może pracować „równocześnie". Unix współpracuje ze specjalnym modułem sprzętowym dokonującym przekształceń adresów i przesyłającym małe bloki programu lub danych, gdy są one potrzebne. Aby zminimalizować liczbę dostępów do dysku, stosuje się algorytm MFU (ang. *most frequently used*, najczęściej używany). Jeśli sprzęt obsługujący pamięć wirtualną jest niedostępny, to Unix musi każdy zawieszony proces przerzucać od razu na dysk, co znacznie zmniejsza efektywność systemu.

To, co widzi się podczas używania Unixa, jest środowiskiem użytkownika. Sprzężenie z poleceniami Unixa i z leżącym wewnątrz właściwym systemem operacyjnym zapewniają różne powłoki (ang. *shell*). Unix zawiera wiele uniwersalnych poleceń do zarządzania plikami, buforowania wydruków (ang. *spooling*), poczty elektronicznej, przetwarzania wsadowego i w tle oraz do administrowania samym systemem. Dla doświadczonych osób polecenia te tworzą mocne, dobrze przystosowywalne środowisko obliczeniowe. W szczególności, Unix odzwierciedla zainteresowania jego twórców i stanowi modelowe środowisko do wytwarzania oprogramowania. Z tych powodów otrzymuje silne poparcie ze strony uczelni i organizacji rządowych.

Unix pozostałby izolowany w środowisku akademickim, gdyby nie był tak łatwo przenośny. Ze względu na to, że został napisany w języku C, daje się przenosić na komputery o różnych architekturach. W konsekwencji, Unix stał się faktycznym standardem wśród systemów operacyjnych, eliminując inne, opracowane indywidualnie przez poszczególnych wytwórców komputerów. Stanowi obecnie stabilną podstawę do wytwarzania oprogramowania użytkowego, niezależnie od specyficznej konfiguracji sprzętowej.

Od DOS-a do Unixa

Ogromna liczba zastosowań mikrokomputerowych opartych na DOS-ie przyciąga olbrzymią liczbę użytkowników. Gdyby nie to, wielu spośród nich prawdopodobnie nie używałoby w ogóle komputerów. Użytkownicy ci mają własne spojrzenie na informatykę przez pryzmat zastosowań. O ile Unix ma swoich systemowych guru, to DOS ma swoich rzemieślników zastosowań.

Użytkownicy DOS-a zmierzają w kierunku maksymalnej użyteczności, tak by być jak najbliżej problemu, który należy rozwiązać. Traktują komputer jako proste w obsłudze i niezawodne urządzenie z danym, sprawdzonym oprogramowaniem usługowym i dwunastoma klawiszami. Jednak użytkownicy dobrze wiedzą, jak oceniać oprogramowanie i konkurujące ze sobą pakiety.

Niestety, wykorzystanie oprogramowania usługowego pracującego pod kontrolą DOS-a nie jest aż tak proste, jak być powinno. Przykładowo, komputer PC, zakupiony do pracy z arkuszami Lotus 1-2-3, będzie z pewnością służył innym programom; wkrótce niezbędne będą także karty graficzne, dodatkowe porty, drukarki, modemy bądź taśmy. Prawdopodobnie trzeba będzie także zainstalować nowe oprogramowanie, skonfigurować urządzenia, wyszukiwać ewentualne usterki, wykonywać regularne składowania, by tworzyć kopie zapasowe cennych plików.

Mimo tego wytwórcy Unixa żywią nadzieję, iż użytkownicy komputerów osobistych „wyrastają" z modeli PC i dlatego chcą ich przyciągnąć do stosowania Unixa. Przykładowo, komputery z Unixem zapewniają:

- podstawy do zastosowań wieloużytkowych, na przykład zarządzanie bazami danych, przetwarzanie tekstów, komunikację,
- scentralizowane przechowywanie danych i mechanizmy ochrony plików, systemowe składowanie kopii rezerwowych plików, rozproszony dostęp dla użytkowników,
- obsługę współużytkowania i dostępu do wielu drukarek, modemów, stacji taśm i terminali,
- scentralizowany system administrowania systemem, który wykonuje regularne nadzorowanie zadań i monitorowanie działania systemu.

Powyższe cechy mają przekonać użytkowników PC, że potrzebują oni systemu operacyjnego o cechach Unixa. Najtrudniej jest przekonać ich, że pod jego kontrolą mogą pracować równie produktywnie jak z DOS-em. Niestety, wielu ludzi zrównuje środowisko użytkowe Unixa z samym systemem operacyjnym Unix. Unix umożliwia tworzenie różnorodnych sprzężeń użytkowych, włącznie ze środowiskiem wielookienkowym. Program usługowy pracujący pod nadzorem Unixa może zapewniać kompletne środowisko, w którym nigdy nie trzeba odwoływać się do Unixa w sposób bezpośrednio widoczny.

Przenoszenie oprogramowania usługowego DOS-a do środowiska Unixa to jedna z dróg zapewniająca utrzymanie znanego sprzężenia i nowe możliwości. Nie zawsze jest to jednak łatwe zadanie, ponieważ nie wszystkie programy są projektowane pod kątem współpracy z systemem operacyjnym na odpowiednio wysokim poziomie, choć jak dotychczas liczba tego rodzaju programów jest już dość znaczna. Pewną możliwość rozwiązania tych problemów daje mikroprocesor Intel 80386, pozwalający bez przeróbek pracować programom usługowym DOS-a pod nadzorem systemu Unix.

Tworzenie wirtualnego komputera PC

Układ Intel 80386 z odpowiednim oprogramowaniem umożliwia tworzenie w środowisku Unixa wirtualnych komputerów PC. Aby zrozumieć jak to działa, warto przyjrzeć się architekturze mikroprocesora Intel 80386. Został on tak zaprojektowany, aby mógł pracować w czterech trybach z łatwym przełączaniem z jednego trybu na drugi.

Domyślnym trybem działania jest tzw. 16-bitowy tryb rzeczywisty. Zapewnia on zgodność z mikroprocesorami Intel 8086 i 8088, które są używane w najstarszych modelach IBM PC. W trybie rzeczywistym Intel 80386 wykonuje zbiór rozkazów 8086. Tak więc kopia programu Lotus 1-2-3 kupiona dla oryginalnego PC będzie poprawnie pracować na mikrokomputerze Compaq 386. Niestety, tryb rzeczywisty nie daje możliwości wykorzystania bardziej zaawansowanych właściwości mikroprocesora.

Intel 80386 ma także tryby ochronne: 16- i 32-bitowy. Tryb ochronny 16-bitowy zapewnia zgodność z mikroprocesorem Intel 80286. Właśnie w tym trybie pracuje system operacyjny OS/2. Ochronny tryb 32-bitowy jest trybem rodzinnym (ang. *native mode*), czyniącym 80386 najpotężniejszym mikroprocesorem Intela. W trybie tym jest zrealizowany mechanizm pamięci wirtualnej ze stronami ściąganymi na żądanie (o wirtualnej przestrzeni adresowej 4 GB) oraz mechanizm przywilejów, dzięki czemu jest możliwa prawdziwa

wielozadaniowość. Tak więc procesor 80386 dobrze pasuje do systemu Unix.

W trybie rzeczywistym programy usługowe DOS-a pracują w środowisku jednozadaniowym, gdzie w razie potrzeby mają dostęp do sprzętu i nie muszą współzawodniczyć z innymi programami. W trybie ochronnym procesor używa poziomu uprzywilejowanego, aby wyznaczyć kolejny proces do wykonania. Tak jak w każdej sytuacji, gdzie zasoby są ograniczone, a potrzeb wiele, musi zostać ustalony pewien priorytet przydziału tych zasobów.

Czwarty tryb pracy 80386 zaprojektowano z myślą o wykonywaniu programów DOS-a w środowisku wielozadaniowym; jest on zwany także trybem wirtualnym 8086, w przeciwstawieniu do trybu rzeczywistego 8086. W trybie rzeczywistym programy DOS-a używają fizycznych adresów pamięci, w trybie wirtualnym natomiast pamięć fizyczna jest przekształcana na przestrzeń adresową pamięci wirtualnej o wielkości 1 MB. Program DOS-a nadal działa tak, jakby miał do czynienia z właściwymi adresami, ale w rzeczywistości są one tłumaczone na adresy wirtualne.

W trybie wirtualnym są wychwytywane także rozkazy wykonujące operacje na sprzęcie, np. rozkazy wejścia-wyjścia. Typowy program DOS-a działa tak, jakby był właścicielem całego komputera, często bezpośrednio manipuluje pamięcią ekranu lub rejestrami sterującymi urządzeniem, zamiast korzystać z pośrednictwa BIOS-a. W systemie wielozadaniowym takim jak Unix rozkazy w rodzaju powyższych są uprzywilejowane. Odpowiednie zasoby muszą zostać przydzielone do zadania i pozostać w jego gestii dopóki rozkazy te nie zostaną bezpiecznie wykonane bez interferencji z innymi procesami.

Po wychwyceniu w trybie wirtualnym rozkazu uprzywilejowanego, mikroprocesor 80386 przełącza się na tryb ochronny. Sam nie realizuje rozkazu, lecz do jego przetworzenia wykorzystuje specjalny program zwany monitorem. Najbardziej znanymi takimi programami są Merge 386 firmy Lotus Computing oraz VP/ix innej amerykańskiej firmy Interactive Systems. Oba programy w pierwszym kroku wykonują tłumaczenie rozkazów 8086 na odwołania systemowe, które mogą być przetwarzane przez Unix. W ten sposób 80386 zapewnia, że program usługowy DOS-a pracujący na wirtualnym komputerze PC chcąc uzyskać dostęp do zasobów systemowych przechodzi przez system operacyjny Unix.

Dokonując odwzorowania adresów fizycznych na wirtualne i wychwytując rozkazy z poziomu sprzętu, mikroprocesor 80386 pozwala programowi DOS-a działać jak na 8088, bez obaw, że spowoduje on interferencje z innymi pogramami lub samym systemem operacyjnym jako całością. Można też wykonywać wiele egzemplarzy DOS-a równocześnie, odnosząc wrażenie, że każdy z nich dysponuje całym komputerem. Emulacja jest tak pełna, że można nawet przeładować komputer wirtualny naciskając kombinację klawiszy *Ctrl-Alt-Del*.

Środowisko wirtualnego DOS-a

Mimo że układ 80386 umożliwia wykonywanie zbioru rozkazów 8086, to jednak nie rozwiązuje to wszystkich problemów z tworzeniem interakcyjnego środowiska DOS-a pod kontrolą Unixa. Monitory Merge 386 i VP/ix realizują jednak takie dodatkowe funkcje jak:

- odwzorowanie systemu plików DOS-a na system plików Unixa, z uwzględnieniem różnicy w konwencjach nazewnictwa plików oraz zapewnieniem dostępu do dysków elastycznych i stałych przy użyciu specyfikacji stacji dysków DOS-a;

- obsługiwanie większości terminali ASCII normalnie dołączonych w systemie Unix, tak aby mogły z nich korzystać programy DOS-a (przykładowo, terminale ASCII mogą wyświetlać 24 wiersze tekstu, zamiast 25 wierszy jak na PC; wielu terminalom brakuje także klawiszy funkcyjnych i specjalnych np. *Alt*),

- odwzorowanie nazw urządzeń DOS-a na nazwy urządzeń Unixa lub nazwy odpowiednich programów realizujących dostęp do tych urządzeń; umożliwia to, na przykład, wysłanie wydruku skierowanego na LPT1 do programu buforującego systemu Unix.

Aby pracować w środowisku DOS-a, przy rejestrowaniu się w systemie trzeba wykazać się posiadaniem konta. Po zarejestrowaniu się można wydać pojedyncze polecenie, po czym ukaże się znany symbol gotowości DOS-a i od tej chwili można już wykonywać polecenia DOS-a, tak jak gdyby pracowało się z prawdziwym systemem DOS.

Można utworzyć środowisko DOS-a na całą lub część sesji pracy z systemem albo utworzyć kopię DOS-a, by uruchomić konkretny program usługowy. Można nawet zainicjować środowisko DOS-a z pliku automatycznego rozpoczęcia sesji (ang. *automatic start-up file*) w taki sposób, że rejestrowanie następuje już bezpośrednio w DOS-ie i nawet nie widać zgłoszenia się Unixa. Pracując pod nadzorem DOS-a można wykorzystywać polecenia Unixa, a także jego pliki.

Zarówno Merge 386, jak i VP/ix odróżniają polecenia DOS-a od poleceń Unixa, dzięki czemu można wykonywać polecenia DOS-a bezpośrednio z Unixa. Używając Unixa, można nadal swobodnie korzystać z programów DOS-a i jego plików, zachowując zalety środowiska wielozadaniowego. Przykładowo, można uruchomić kilka programów usługowych DOS-a w tle oraz wykonywać wiele programów DOS-a i Unixa współbieżnie.

Efektywność środowiska wirtualnego DOS-a może zmieniać się w zależności od obciążenia systemu (liczby użytkowników i rodzaju prac, które wykonują) i charakteru programu usługowego. Przykładowo, przeliczanie arkusza elektronicznego może być szybsze niż na PC, gdyż zależy tylko od szybkości procesora. Inne zadania mogą zawierać dużo operacji na poziomie sprzętowym, które muszą być wychwytywane, tak więc efektywność ich wykonywania będzie mniejsza.

Problemy zgodności

Wśród programów, które mogą pracować dzięki zastosowaniu Merge 386 lub VP/ix znajdują się między innymi Lotus 1-2-3, VisiCalc, Multiplan, WordStar, Word, WordPerfect. Merge 386 i VP/ix umożliwiają pracę większości programów DOS-a, włączając w to programy korzystające z metod bezpośredniego dostępu do sprzętu. Niektóre programy usługowe DOS-a wykorzystują dodatkowo zależności czasowe, zgrane z czasami odpowiedzi sprzętu. Takie oprogramowanie w środowisku wielozadaniowym czasami zachowuje się w sposób kłopotliwy.

Kolejnym problemem do rozwiązania przy przenoszeniu programów usługowych z PC do środowiska Unixa jest jakość monitora. Standardowa konsola komputera PC ma monitor graficzny odwzorowujący pamięć: program wpisuje odpowiednie dane do zarezerwowanego obszaru pamięci, która jest odwzorowywana na ekranie. Jednak większość terminali systemów pracujących z Unixem to terminale ASCII. Na szczęście prawie wszystkie programy DOS-a mogą pracować z kartą monochromatyczną MDA, działającą w trybie znakowym, tak jak terminal ASCII. Większość z nich może więc pracować na terminalu ASCII emulującym kartę monochromatyczną MDA.

Jednak nawet przy użyciu karty MDA pojawiają się pewne kłopoty: zbiór znaków wyświetlanych na PC obejmuje znaki specjalne wykraczające poza standard ASCII. Większość terminali ASCII nie może wyświetlać tych znaków. Merge 386 wyświetla je jako gwiazdkę *, a VP/ix pozwala definiować własne odpowiedniki tych znaków przez odpowiedni zapis w pliku opisu terminala.

Można tego uniknąć, jeśli dysponuje się terminalem umożliwiającym wyświetlanie rozszerzonych kodów znaków PC (ang: *PC-scancode terminal*) i emulującym kartę MDA. Wyświetla on pełnych 25 wierszy wymaganych na ekranie PC i rozszerzony zestaw znaków (semigrafikę). Chociaż może rozwiązać to problemy związane z programami usługowymi pracującymi w trybach znakowych, to jednak programy graficzne, jak np. Flight Simulator firmy Microsoft, będą pracowały tylko na monitorach działających na zasadach odwzorowywania pamięci.

Monitor działający na zasadzie odwzorowywania pamięci wymaga przesyłania większej liczby danych niż jest to możliwe przez łącze szeregowe. Aby osiągnąć duże szybkości przesyłania danych, trzeba dołączyć konsolę PC bezpośrednio do procesora i pamięci przez bardzo szybką magistralę. Terminal firmy Sun River odwzorowujący pamięć został zaprojektowany pod kątem pracy z systemami wieloużytkowymi, z programami graficznymi DOS-a. Jest on przyłączony do systemu przez szybkie, światłowodowe łącze zamiast przez kabel RS232.

Jednym z systemów opartych na 80386 umożliwiających pracę programów graficznych DOS-a jest stacja robocza Sun 386i firmy Sun Microsystem. Wykonując swoją własną

ulepszoną wersję VP/ix, zwaną DOS Windows, umożliwia ona pracę monitorów grafiki kolorowej i monochromatycznej z rozdzielczością do 1152 × 900. Oprogramowanie emuluje tryby pracy: monochromatyczny, Herculesa i grafiki kolorowej. Możliwe jest także stosowanie grafiki EGA i VGA. Program usługowy DOS-a pracuje w stacji Sun 386i w swoim własnym oknie o stałych rozmiarach, stanowiącym tylko część pełnowymiarowego monitora. Można, przynajmniej dla celów demonstracyjnych, mieć wiele okien, w których w tym samym czasie pracują na przykład programy Flight Simulator i Auto-CAD.

Stacja Sun 386i jest dostarczana z oprogramowaniem umożliwiającym pracę z Microsoft Windows. Programy usługowe pracujące z tym oprogramowaniem mogą wykorzystywać pełny rozmiar ekranu stacji Sun 386i.

* * *

Możliwości DOS-a zapewnione przez układ Intel 80386 i wspomagane przez Merge 386 i VP/ix mają zachęcić użytkowników DOS-a do przechodzenia na system Unix. Aby zachęta była skuteczna, użytkownicy DOS-a muszą być przekonani o wartości Unixa i o długiej perspektywie jego użyteczności.

To, co jest naprawdę potrzebne, to pomost DOS–Unix. Wykonywanie programów usługowych DOS-a pod nadzorem systemu Unix nie jest najlepszym rozwiązaniem z punktu widzenia projektów. Jeśli jednak twórcy oprogramowania na DOS-a stwierdzą, że użytkownicy wykazują chęć przejścia na Unixa, być może zachęcą się do opracowywania oprogramowania użytkowego dla Unixa, zgodnego z wersjami opracowanymi dla DOS-a.

Oprac. **D. GRABOWICZ**
Na podstawie Byte
IBM Special Edition Fall 1988

Kto jest kim

Mike G. Rodd

Przewodniczący Komitetu Technicznego IFAC ds. Komputerów, prof. Rodd, otrzymał kolejne stopnie naukowe, włącznie z tytułem doktora w zakresie elektrotechniki, na Uniwersytecie Kapsztadzkim w Południowej Afryce. Po pięciu latach pracy w przemyśle w dziedzinie sterowania, został starszym wykładowcą elektrotechniki na Uniwersytecie w Kapsztadzie. Tam też był jednym z twórców jednostki badawczej współpracującej z przemysłem i zajmującej się wykorzystaniem techniki mikroprocesorowej w przemyśle przetwórczym i hutnictwie.

W 1979 r. został profesorem elektrotechniki na Uniwersytecie Witwatersrand w Johannesburgu. Stworzył tam finansowany przez przemysł program badawczy Mechatronics, poświęcony pracom nad wprowadzeniem nowoczesnych metod produkcyjnych. W 1986 r. Mike Rodd został profesorem elektrotechniki i elektroniki, jak również szefem wydziału nauk komputerowych Uniwersytetu Walijskiego w Swansea.

Naukowe zainteresowania prof. Rodda obejmują zagadnienia sterowania w czasie rzeczywistym rozproszonych systemów komputerowych (ze szczególnym uwzględnieniem procesów przemysłowych), systemy komunikacyjne oparte na modelu OSI, sieci lokalne na potrzeby przemysłu, rozpoznawanie obrazów jako narzędzie kontroli, sterowniki o programowalnej logice i zbieranie danych w czasie rzeczywistym. Interesuje go również wykorzystanie metod sztucznej inteligencji do rozwiązywania problemów występujących w wytwarzaniu i sterowaniu. Jest autorem 75 publikacji z tych i pokrewnych dziedzin.

Prof. Rodd jest członkiem brytyjskich organizacji IEE i IMC (*Institution of Electrical Engineers, Institute of Measurement and Control*), a także – amerykańskiego instytutu IEEE, będąc delegatem grupy ds. wytwarzania do komitetów AMT BSI (*British Standards Institution*). Jest również redaktorem czasopisma Journal for Engineering Applications of Artificial Intelligence.

W działalności pozazawodowej prof. Rodd, jeszcze jako student, był przewodniczącym Studenckiej Organizacji Ochrony Zdrowia Uniwersytetu Kapsztadzkiego. Organizacja zajmowała się niesieniem pomocy w najbardziej zaniedbanych rejonach Kapsztadu i okolic. Przez sześć lat był też ochmistrzem w domu dzieci specjalnej troski. Zainteresował się ruchem Czerwonego Krzyża i pełnił kilka funkcji w Czerwonym Krzyżu Afryki Południowej.

MK
na podstawie IFAC Newsletter

Tomografia komputerowa

dokończenie ze s. 14

Ważnym miernikiem jakości rekonstruowanego obiektu jest jego rozdzielczość przestrzenna i gęstościowa. *Rozdzielczość przestrzenna* jest związana ze zdolnością systemu do rozróżniania struktur o dobrym kontraście (powietrze, kość), umieszczonych wewnątrz jednorodnego tła (np. w wodzie), zależy więc od rozmiarów detektorów. Wyraża się ją przez rozmiary obrazu, a więc również rozmiary pola odpowiadającego jednemu elementowi obrazu – pikselowi. Przykładem mogą być tutaj tomografy IV generacji, za pomocą których rekonstruowane są obrazy dwuwymiarowe o rozmiarach 512 × 512 pikseli, gdzie piksel reprezentuje obszar o powierzchni 0,5 × 0,5 mm². *Rozdzielczość gęstościowa* jest natomiast związana ze zdolnością systemu tomograficznego do rozróżniania struktur o zbliżonej gęstości. Jest ona miarą dokładności pomiaru dla każdego elementu rekonstruowanego obrazu, zależy więc od szumów pomiarowych oraz niejednorodności i ziarnistości obrazu.

* * *

Tomografia komputerowa stosowana w medycynie rozwija się obecnie w trzech kierunkach:

- zmniejszenie czasu badania i dawki promieniowania, jaką otrzymuje pacjent podczas badań,
- udoskonalenie technik badania fizjologii dynamicznych narządów wewnętrznych człowieka (serce, płuca),
- zwiększenie dokładności rekonstrukcji.

Rozwiązanie tych problemów próbuje się osiągnąć przez poprawianie jakości zbierania danych pomiarowych (zmiany w konstrukcji skanerów oraz wprowadzanie różnego typu korekcji zmniejszających błędy pomiarowe), a także przez rozwój algorytmów i architektur równoległych, których zastosowanie może diametralnie zminimalizować czas rekonstruowania i wizualizacji. Architektury takie mogą być wykorzystane zarówno do implementacji algorytmów rekonstruowania, jak również do przetwarzania i wizualizacji zrekonstruowanych obrazów.

LITERATURA

[1] Bates R. H. T., Garden K. L., Peters T. M.: Overview of computerized tomography with emphasis on future developments. Proc. IEEE, Vol. 71, No. 3, March 1983
[2] Herman G. T., Hsun Kao Liu: Three-Dimensional Display of Human Organs from Computed Tomograms. Computer Graphics and Image Processing, No. 9, pp. 1–21, 1979
[3] Jaman K. A., Gordon R., Rangayyan R. M.: Display of 3D Anisotropic Images from Limited-View Computed Tomograms. CVG-IP, Vol. 30, pp. 345–361, 1985
[4] Macovski A.: Physical Problems of Computerized Tomography. Proc. IEEE, Vol. 71, No. 3, March 1983
[5] Mokrzycki W.: Techniki obrazowania informacji o trójwymiarowych zdyskretyzowanych scenach. Informatyka, nr 7, 1988
[6] Nowiński W. L.: Rekonstruowanie obrazów w tomografii komputerowej. Informatyka, nr 3, 1989
[7] Nowiński W. L.: Algorytmy rekonstruowania obrazów za pomocą metod transformacyjnych. Informatyka, nr 4, 1989
[8] Nowiński W. L.: Algorytmy rekonstruowania obrazów metodami rozwinięcia w szereg. Informatyka, nr 5, 1989
[9] Robb R. A. et al.: High-Speed Three-Dimensional X-Ray Computed Tomography – The Dynamic Spatial Reconstructor. Proc. IEEE, Vol. 71, No. 3, March 1983.

ambex®

ZAKŁAD ELEKTRONIKI

JEDNOSTKA INNOWACYJNO-WDROŻENIOWA

02-770 Warszawa, ul. Żabińskiego 7, tel. 24-15-69

oferuje

idealne do Twojego IBM PC XT/AT
oparte na elementach najwyższej światowej klasy

PRZETWORNIKI ANALOGOWO/CYFROWE i CYFROWO/ANALOGOWE 12-BITOWE

- precyzyjne wzmacniacze próbkująco-pamiętające
- pomiar analowy jednoczesny na wielu kanałach
- czasy przetwarzania przetworników a/c od 1 do 50 μs
- przetworniki a/c, c/a i we/wy sterujące na jednej płytce

SYSTEMY POMIAROWE „POD KLUCZ"

- tworzone przy udziale wybitnych specjalistów z różnych dziedzin
- IBM PC XT/AT z wysokiej klasy przetwornikami a/c i c/a
- wzmacniacze pomiarowe z izolacją galwaniczną sygnałów
- oprogramowanie zgodnie z wymaganiami klienta

WZMACNIACZE POMIAROWE

WZMACNIACZE Z IZOLACJĄ GALWANICZNĄ SYGNAŁÓW ANALOGOWYCH

OPROGRAMOWANIE SPECJALIZOWANE

UWAGA: naszym klientom zapewniamy bezpłatne oprogramowanie standardowe, wszechstronne konsultacje w okresie użytkowania sprzętu oraz serwis gwarancyjny i pogwarancyjny.

TECHNIKA POMIAROWA, STEROWANIE, POMIARY MEDYCZNE I LABORATORYJNE

EO / 276 / 89

Architektura dla Ady

O ile przemysł elektroniczny przeszedł znaczną ewolucję, od próżniowych lamp elektronowych do układów o bardzo dużym stopniu scalenia, to skomplikowane systemy oprogramowania są wykonywane wciąż metodami tradycyjnymi, znanymi z pierwszych dni rozwoju komputeryzacji. To opóźnienie w rozwoju oprogramowania stało się problemem krytycznym. Kluczem do jego rozwiązania jest zastosowanie zasad nowoczesnej inżynierii oprogramowania w połączeniu z wyrafinowanymi językami wysokiego poziomu oraz narzędziami wspierającymi i wymuszającymi stosowanie tych zasad i języków.

W 1980 r. amerykańska firma Rational rozpoczęła prace nad środowiskiem projektowo-uruchomieniowym dla Ady nazwanym *Rational Environment*, dostarczającym obszernego zestawu narzędzi dla projektantów skomplikowanych systemów pisanych w Adzie. Zastosowanie narzędzi zwiększa efektywność poszczególnych programistów i całego zespołu, jak również poprawia jakość tworzonego oprogramowania. Środowisko to, wykorzystywane praktycznie od 1985 r. w ponad 30 instalacjach, stanowi dużą pomoc przy rozwiązywaniu problemów składniowych i semantycznych, zawiera programy uruchomieniowe, zarządzania konfiguracją, kontroli wersji źródłowej, śledzenia historii i inne narzędzia uniwersalnego środowiska macierzystego.

Jego istotną cechą jest możliwość wykonywania kompilacji przyrostowej jednostek programowych Ady. W zwykłych środowiskach, istniejący cykl edycji, kompilowania, uruchamiania i ponownej edycji znacznie opóźnia postępy projektu, scalanie i testowanie bardzo skomplikowanych systemów. W środowisku Rational Environment można dokonywać zmian przyrostowo używając pojedynczych deklaracji i instrukcji Ady, dzięki czemu znacznie zmniejsza się czas potrzebny na przetestowanie złożonego systemu.

Podstawowym zadaniem środowiska jest umożliwienie pojedynczemu programiście pracy o wysokim stopniu interakcyjności, przy jednoczesnym zapewnieniu środków do koordynacji pracy całego zespołu. Aby osiągnąć ten cel, postanowiono użyć Ady jako języka implementacji, wykorzystując szczególnie ścisłą typizację języka, wielozadaniowość, tworzenie pakietów i jednostek rodzajowych (pakiety są niezbędne do dekompozycji złożonych systemów, a jednostki rodzajowe umożliwiają stosowanie wielokrotnie używalnych składników oprogramowania). Dzięki napisaniu od 1980 r. ponad dwóch milionów wierszy kodu Ady, firma Rational uzyskała duże doświadczenie w tworzeniu skomplikowanych systemów w Adzie, co wpłynęło bardzo pozytywnie na funkcjonalność utworzonego środowiska.

Równie ważnym czynnikiem osiągnięcia założonego celu był wybór Diany (ang. *Descriptive Intermediate Attributed Notation for Ada*) jako języka pozwalającego przedstawić program w Adzie w postaci struktury drzewiastej i zakodować kompletne informacje o składni i semantyce każdej jednostki programowej Ady. Użycie Diany jako mechanizmu integrującego środowisko umożliwia prowadzenie wszechstronnej edycji, uruchamiania i badania programów – czynności stanowiących podstawy kompilacji przyrostowej.

Firma Rational kontrolowała tworzenie tak skomplikowanego środowiska projektowo-uruchomieniowego za pomocą szybkiego makietowania i szeroko stosowanej symulacji. Większą część tej pracy wykonano na komputerze DEC 2060 i wielu maszynach Data General MV. Szybko okazało się jednak, że konwencjonalne architektury nie mogą zapewnić wydajności potrzebnej do spełnienia głównego celu środowiska – interakcyjności i integralności. Z tego powodu firma musiała podjąć decyzję, czy zredukować zakres (funkcjonalność) środowiska, czy znaleźć architekturę odpowiednią do jego uruchomienia. Ostatecznie zdecydowano się zaprojektować od podstaw własną maszynę – Rational – Ri000. Decyzja ta nie wynikała oczywiście z trudności zaimplementowania Ady na sprzęcie o konwencjonalnej architekturze, ale z wymagań stawianych projektowanemu środowisku.

Ri000 jest więc specjalizowaną maszyną Ady , zoptymalizowaną do potrzeb interakcyjnego środowiska programowego. Zamiast przyjęcia architektury opartej na liście rozkazów, jak w podejściu klasycznym, Ri000 oparto na modelu wykonawczym programu Ady. Tak więc wprowadzono pojedyncze rozkazy maszynowe do wykonywania takich operacji, jak konkretyzacja jednostki rodzajowej i uaktywnienie zadania. Spotkania zadań zostały tak zoptymalizowane, że nakłady obliczeniowe potrzebne do ich realizacji są porównywalne z wywołaniami procedur w maszynach tradycyjnych. W celu zwiększenia szybkości zastosowano szerokie magistrale – jednostka centralna działa na 128-bitowych słowach, a mikrokod ma słowa 203-bitowe. Wykorzystano też duży system pamięci wirtualnej o 67-bitowych adresach.

Osiągnięcie dużego stopnia interakcyjności i integralności środowiska jest nie tylko sprawą zwiększenia możliwości sprzętowych. Główną przeszkodą są konsekwencje nowej koncepcji oprogramowania środowiska, stojącej w sprzeczności ze zwykle przyjmowanymi założeniami o zachowaniu się programu w czasie wykonania. Użycie Diany spowodowało konieczność zastosowania dużej, wirtualnej przestrzeni adresowej. Wszystkie jednostki Ady są przechowywane w formacie Diany. Choć reprezentacja taka przynosi wiele niepowtarzalnych korzyści, np. umożliwia kompilację przyrostową, jest bardzo pamięciochłonna, gdyż stosunek kodu źródłowego do reprezentacji w Dianie wynosi 1:8. Ponieważ Ada jest wykorzystywana jako język poleceń i język implementacyjny, z różnych względów trudno zapewnić lokalność odwołań w czasie istnienia pojedynczego procesu.

W środowisku Rational można również wykonywać szybkie kartkowanie programów Ady (ang. *browse*). Przykładowo, użytkownik może wskazać obiekt i w szybko wyświetlonym oknie uzyskać opis zdefiniowanego wystąpienia tego obiektu. Efektywność dostępu do pamięci wymagana przez takie interakcyjne, semantyczne zapytania osiągnięto przez asocjacyjną organizację głównej pamięci 32 MB. Dodatkowo, w Rational Environment wykorzystano wielozadaniowość Ady do zarządzania współbieżnością, a obsługę wyjątków jako modelu wykrywania i usuwania defektów. Rozróżnianie typów przesunięto do czasu wykonania, co umożliwia kompilację przyrostową, współdzielenie kodu rodzajowego i dokonywanie zmian w ciele jednostek rodzajowych bez późniejszej konkretyzacji. W wypadku braku sprzętu o odpowiedniej architekturze nie można by stworzyć środowiska o takim stopniu interakcyjności.

Cechę tę osiągnięto bez posługiwania się szczególnie rewelacyjnymi technikami implementacyjnymi. Jednostka centralna jest zbudowana z układów logiki TTL Schottky'ego w wersji ulepszonej. Nie zastosowano mikroukładów wykonywanych na zamówienie, chociaż procesor ma na każdej płycie jeden mikroprocesor do diagnostyki w czasie rzeczywistym. Tak więc, podstawową nowością wprowadzoną w Ri000 jest jego architektura, bezpośrednio odpowiadająca budowie interakcyjnego środowiska programowego Ady.

Opracował M. KUC
na podstawie Computer Design

Bardzo prosimy Autorów przysyłających teksty artykułów do opublikowania w INFORMATYCE o podawanie danych umożliwiających bezpośredni kontakt w godzinach pracy (numer telefonu) oraz adres domowy (do korespondencji i wysłania honorarium).

Pogromca konkurentów – dBase IV

W obliczu silnej konkurencji ze strony rywalizujących systemów zarządzania bazami danych, takich jak Paradox i R:base, firma Ashton Tate wprowadzając swój nowy program dBase IV ponownie próbuje dowieść swego pierwszeństwa na rynku oprogramowania mikrokomputerowych baz danych.

Wyposażony w zasadniczo nowe cechy, takie jak obsługa języka SQL (ang. *Structured Query Language*), kompilator kodu źródłowego oraz istotnie poprawiony sprzęg z użytkownikiem, dBase IV wydaje się sprostać swoim głównym konkurentom, a nawet przewyższać ich możliwościami, zachowując przy tym pełną kompatybilność z dBase III. Producent utrzymuje bowiem, że wszystkie bazy danych, formularze, raporty i zastosowania opracowane w dBase III będą działać bez żadnej modyfikacji w dBase IV.

Firma Ashton Tate traktuje dBase IV jako pierwszy, lecz rozstrzygający krok w kierunku realizowania całkowitej łączności między komputerami osobistymi, tzw. środowiskami grup roboczych (ang. *workgroup environments*) i SQL-owymi bazami danych na dużych komputerach (ang. *mainframe*). Jednak, mimo że dBase IV obsługuje pełny zestaw poleceń SQL-a zarówno w trybie interakcyjnym jak i polecenia zanurzone w kodzie programowym dBase, nie przewidziano przenoszenia danych między dBase a SQL-owymi bazami danych. Implementacja SQL-a w dBase IV jest więc pożyteczna tylko dla programistów, którzy chcieliby pisać kod z tą myślą, aby go potem przenieść do środowiska SQL-a.

Prawdziwa przenośność baz danych między środowiskami SQL-a i dBase IV zostanie chyba zrealizowana dopiero w planowanym dla systemu OS/2, a zapowiedzianym wspólnie przez firmy Microsoft i Sybase, oprogramowaniu obsługi SQL-a – SQL Server. Ostatecznym celem firmy Ashton Tate jest natomiast uczynienie z dBase bazy danych do wyboru dla tzw. usługodawcy plików (ang. *file server*) w systemach Unix i VMS, jak również w systemach MS-DOS i OS/2.

Pożegnanie z przeszłością

Stanowiąc duży krok naprzód w technologii baz danych, dBase IV oznacza także zasadnicze zerwanie z przeszłością. Dotychczas dBase miał zawsze reputację oprogramowania trudnego do opanowania i trudnego w użyciu, gdyż wymagał znacznego wysiłku przy programowaniu. Na te zarzuty firma Ashton Tate odpowiedziała całkowitym przeprojektowaniem sterowanego przez menu sprzęgu użytkownika (chociaż nadal można posługiwać się znanym mechanizmem podpowiedzi kropkowych, ang. *dot prompts*), poprawieniem generatora zastosowań, umożliwieniem makrowywołań z klawiatury oraz udogodnieniem zwanym *zapytanie przez przykład* (ang. *query-by-example*, QBE), podobnym do zaimplementowanego w Paradoxie.

Ponadto, nowe generatory raportów i formularzy zawierają mechanizm menu do projektowania plansz ekranowych, pól wyliczeniowych, sprawdzania danych, rysowania ramek i linii oraz inne nowe udogodnienia do przygotowywania wejściowych i wyjściowych plansz ekranowych. Nowy dBase współdziała także z oprogramowaniem graficznym Chart-Master firmy Ashton Tate, do automatycznego generowania rysunków i wykresów. Jednak użycie programu Chart-Master jest możliwe tylko poza systemem dBase.

Struktura bazy danych w dBase IV jest w pełni zgodna ze strukturą wcześniejszych wersji dBase, przy czym dodatkowo wprowadzono nowy zmiennoprzecinkowy typ danych o 64 bitach. System dBase IV pozwala także operować danymi w polach tekstowych memo w taki sam sposób jak danymi w innych polach, co nie było możliwe w dBase III.

Kolejnym wzmocnieniem struktury bazy danych jest możliwość łączenia wielu baz danych w perspektywę (ang. *view*) lub pseudobazę danych, która pod względem funkcjonalnym niczym nie różni się od zwykłej bazy danych. Oznacza to, że operując odpowiednimi formularzami, raportami i kodem aplikacyjnym odwołującym się do perspektywy, można automatycznie wydobywać i aktualizować dane należące do wszystkich baz danych dołączonych do tej perspektywy. Dzięki temu operacje relacyjne są znacznie mniej złożone niż we wcześniejszych wersjach dBase. Jednakże w danej chwili czasu, o ile nie pracuje się w trybie programowym, można mieć do czynienia tylko z jedną perspektywą lub bazą danych.

Nowy dBase umożliwia także automatyczne indeksowanie (do 47 indeksów na jeden plik z bazy danych). Chociaż nadal można tworzyć swoje własne pliki indeksowe NDX, automatyczne indeksowanie w większości wypadków eliminuje potrzebę korzystania z tych plików.

Pakiet dBase IV zawiera również oprogramowanie uzupełniające dla systemowego języka poleceń. Obejmuje ono bibliotekę funkcji finansowych i matematycznych, tzw. „język szablonów" do przystosowywania generatora zastosowań, polecenia do tworzenia okienek z trawersowalnym w górę i w dół menu (ang. *pop-up, pull-down*) oraz nowy wielookienkowy program wspomagający usuwanie błędów. Okienka umożliwiają jednocześnie oglądanie wielu baz danych.

Kompilator kodu źródłowego

Pakiet dBase IV zawiera pewien rodzaj kompilatora. Ten nowy kompilator o nazwie dBase4 wytwarza kod wynikowy, dzięki któremu – według opinii firmy Ashton Tate – programy wykonują się do dziewięciu razy szybciej niż w dBase III. Kompilator tworzy pliki o rozszerzeniu DBO, a nie – COM lub EXE, jak większość innych kompilatorów. Specjalne narzędzie nazwane *Build* pozwala na konsolidowanie programów.

Wreszcie, dBase IV zawiera ulepszenia do zastosowań wielożytkowych. Obejmują one przetwarzanie transakcji, naprawę w wypadku błędu, zabezpieczenia hałasowe i szyfrowe oraz automatyczne blokowanie na poziomie plików i rekordów, z identyfikacją stanowiska komputerowego blokującego plik lub rekord. Pakiet dBase IV działa zarówno w systemie MS-DOS jak i OS/2. Wersja MS-DOS wymaga pamięci RAM o pojemności 640 KB, przy czym automatycznie wykorzystuje istniejące rozszerzenie pamięci.

W pierwszym okresie na rynku będą dostępne dwie wersje dBase IV – wersja standardowa za 795 dolarów dla bieżącej eksploatacji oraz wersja droższa (1295 dolarów) tzw. *Developer's Edition*, przeznaczona dla poważnych programistów i ośrodków rozwojowych. Obie wersje są takie same, przy czym *Developer's Edition* zawiera dodatkową dokumentację oraz narzędzia do łączenia skompilowanych kodów źródłowych, wspomagania pracy w czasie rzeczywistym oraz testowania zastosowań wielożytkowych w lokalnych sieciach komputerowych.

Opracował **WIKTOR RZECZKOWSKI**
na podstawie Byte nr 4, 1988

Wychodząc naprzeciw zapotrzebowaniu środowiska redakcja nasza oferuje łamy INFORMATYKI na bezpłatne zamieszczanie anonsów o imprezach informatycznych organizowanych na terenie kraju.

Zwracamy się do organizatorów konferencji, wystaw, konkursów i sympozjów z prośbą o przysyłanie do redakcji krótkich (pół strony maszynopisu) informacji o planowanych imprezach. Teksty powinny zawierać datę i miejsce imprezy, krótką charakterystykę jej tematyki oraz adres organizatorów.

Oczekujemy na zgłoszenia.

„Bazy danych – od koncepcji do realizacji"

W znanej serii PWE „Informatyka w praktyce" ukazała się pod koniec 1988 r. książka autorstwa pani profesor Colette Rolland dotycząca baz danych [1]. Istotny wkład do tej pracy wnieśli także jej polscy tłumacze: dr Jerzy Marcinkiewicz z Uniwersytetu Szczecińskiego i doc. Jan Stępniewski z Akademii Ekonomicznej we Wrocławiu. Są oni autorami wprowadzenia i części czwartej ilustrującej, przedstawioną w poprzednich częściach tej książki, metodę projektowania baz danych. Jest to w istocie praca poświęcona projektowaniu zastosowań systemów z bazami danych i w tym też zarazem tkwi jej walor szczególny – przesądzający o użyteczności praktycznej, a także przydatności w dydaktyce. Warto też wskazać na to, że praca powstała jako tłumaczenie maszynopisu. Można zatem uznać, że zawiera treść stosunkowo nową. Tak jest lub raczej było w istocie, bo ze względu na długi krajowy cykl wydawniczy książka nieco się postarzała.

Wydanie tej cennej pracy w Polsce było możliwe dzięki prowadzonej od wielu lat współpracy naukowej Instytutu Cybernetyki Ekonomicznej i Informatyki Uniwersytetu Szczecińskiego z panią profesor Colette Rolland. Właśnie w wyniku tej współpracy powstała część czwarta książki opracowana przez autorów polskich.

W książce, co jest szczególnie istotne, położono nacisk na dwa aspekty realizacji baz danych: modelowanie konceptualne rzeczywistości i modelowanie struktur logicznych baz danych. Pominięto natomiast aspekt realizacji fizycznej – zależnej w dużej mierze od sprzętu. Pozwoliło to skoncentrować uwagę głównie na problematyce, która dotyczy strony funkcjonalnej zastosowania – jego przystosowania do wymagań użytkownika.

Znając także dalsze prace C. Rolland oraz współautorów, mogę stwierdzić, że książka stanowi jakby część pierwszą pewnej ogólnej metody realizacji baz danych. Jest ona metodyczną podstawą dla części drugiej, w której główny nacisk kładzie się na aspekty dynamiczne i rozwój systemu informacyjnego. Konsekwencją tego podejścia jest oryginalna metoda dynamiki systemów informacyjnych REMORA przedstawiona w kolejnej pracy C. Rolland napisanej w języku francuskim pt. *„Conception d'un systeme d'information. Methode REMORA"*. Wydaje się, że metoda REMORA w Polsce (także dzięki konferencji INGOFRYF) jest coraz lepiej znana. Dzięki swojej prostocie i formule ukierunkowanej na dynamikę może stać się wiodącym podejściem w dziedzinie konceptualizacji systemów informacyjnych. Dlatego wkrótce należy chyba oczekiwać polskiego wydania także i tej nowej pracy.

Podsumowując, uważam że omawiana książka jest pozycją wartościową, zawiera zwartą i przemyślaną metodologię realizacji systemów z bazami danych. Jej język jest klarowny, terminologia precyzyjna (co jest także zasługą tłumaczy). Książka może być przydatna dla użytkowników systemów informacyjnych, projektantów baz danych i studentów. Może stanowić materiał do inspiracji dalszych badań – szczególnie implementacji komputerowej.

[1] C. Rolland: Bazy danych od koncepcji do realizacji. PWE, Warszawa, 1988

WOJCIECH OLEJNICZAK

digitex

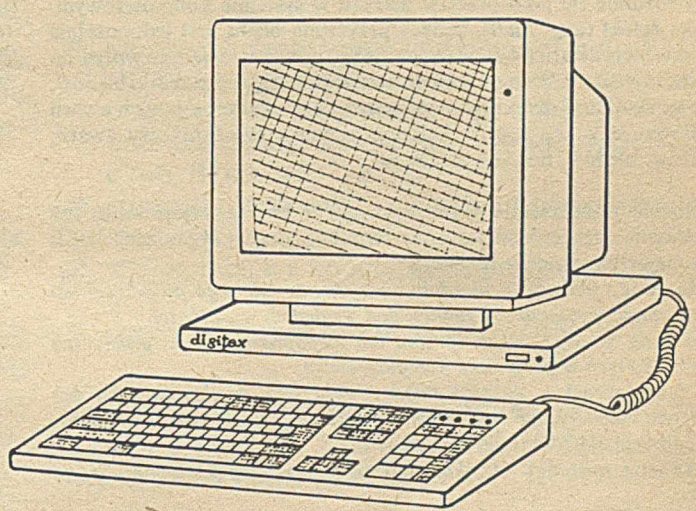
ZAKŁAD SYSTEMÓW CYFROWYCH

81-832 Sopot, ul. Mickiewicza 20
tel. 51-28-27, tlx 512290 dgtex pl

KOLOROWY TERMINAL GRAFICZNY DXT–125

- emuluje protokół terminali graficznych DEC*– a VT – 125*
- emuluje protokół terminali alfanumerycznych DEC-a VT–100*, VT–52*
- przeznaczony do pracy w konfiguracjach wielodostępnych z komputerami serii SM, PDP11, PC XT/AT pod kontrolą systemów operacyjnych **XENIX, UNIX, RSX, MULTILINK** i innych
- wyświetla informacje nie tylko w postaci znaków alfanumerycznych, ale również w postaci wykresów, rysunków itp., zgodnie z protokołem graficznym **ReGIS***, który m. in. zapewnia kreślenie prostych, okręgów, elips, krzywych, zapełnianie, „dostęp" do każdego punktu itp.

- rozdzielczość grafiki:
  512 × 256 punktów – DXT–125A
  640 × 400 " – DXT–125B
  640 × 480 " – DXT–125C
  800 × 480 " – DXT–125D
- atrybuty: 16 kolorów każdego punktu (lub 8 kolorów i migotanie)
- tryb alfanumeryczny: maks. 60 wierszy po 100 znaków
- współpracuje z dowolną drukarką (złącze równoległe lub szeregowe)
- klawiatura typu PC–XT
- monitor kolorowy 14" (RGB standard, EGA lub Multisync – zależnie od rozdzielczości terminala)



*DEC, VT, ReGIS są zastrzeżonymi znakami firmowymi Digital Equipment Co.

EO/471/89

Czym jest rzetelność systemu komputerowego?

Od niedawna, w obcojęzycznej literaturze dotyczącej badania niezawodności pojawia się nowy termin – *dependability*. Do autorstwa tej nazwy przyznaje się grupa robocza 10.4 IFIP-u zajmująca się niezawodnością i tolerowaniem defektów (*IFIP Working Group* 10.4 – *Reliable Computing and Fault Tolerance*). Według definicji [1] termin ten oznacza *właściwość systemu komputerowego, która umożliwia uzasadnione poleganie na usługach zapewnianych przez ten system* (ang. *the property of a computer system that allows reliance to be justifiably placed on the service it delivers).*

Rozszyfrowując tę właściwość trzeba zwrócić uwagę na jej dwa aspekty. Po pierwsze, poleganie na usługach zapewnianych przez system komputerowy jest rozumiane tradycyjnie jako skuteczne wykonywanie przezeń swoich funkcji, a więc wykonywanie niezawodne. Po drugie, poleganie na usługach systemu musi być rozumiane bardziej nowocześnie jako niespełnianie funkcji nie zaplanowanych, co można ująć jako zasadę spełniania swoich funkcji w sposób bezpieczny (dla otoczenia). Tak więc, właściwość zdefiniowana w powyższy sposób ma dwa aspekty – określa, co system powinien robić, a więc jego **niezawodność** (ang. *reliability*), i czego nie powinien robić, tj. jego **bezpieczeństwo** (ang. *safety*) dla otoczenia. Taką właściwość, przejawiającą się na zewnątrz przez niezawodność systemu komputerowego i jego bezpieczeństwo dla otoczenia, można nazwać po polsku **pewnością** lub **rzetelnością.**

Dodatkowym aspektem tej właściwości, niejako implikowanym przez bezpieczeństwo systemu dla otoczenia, jest **zabezpieczenie** samego systemu komputerowego, określane po angielsku mianem *security*. Ten aspekt wyróżniają zarówno eksperci IFIP WG10.4, jak i autorzy recenzowanej w tym numerze książki *Dependability of Critical Computer Systems* (s. 4) – członkowie Komitetu Technicznego nr 7 EWICS, o znamiennej nazwie *Technical Committee on Reliability, Safety and Security*. Wymienione terminy zdefiniowano w tej publikacji następująco:
niezawodność (ang. *reliability*) – *prawdopodobieństwo, że jednostka funkcjonalna będzie wykonywać wymaganą funkcję w określonych warunkach przez określony czas,*
bezpieczeństwo (ang. *safety*) – *zapewnienie uniknięcia zagrożenia lub ochrony przed zagrożeniem dla życia lub ciała, środowiska lub własności, wynikającym z awarii systemu komputerowego,*
zabezpieczenie (ang. *security*) – *ochrona sprzętu komputerowego lub oprogramowania przed przypadkowym lub złośliwym dostępem, użyciem, zmodyfikowaniem lub zniszczeniem.*

W drugiej z tych definicji występuje ważne pojęcie **awarii** (ang. *failure*) zwanej też po polsku **uszkodzeniem,** warto więc podać, jak jest tu rozumiane. W cytowanym artykule [1] stwierdzono, że awaria jest uważana za przejaw błędu w zapewnianej usłudze. **Błąd** natomiast (ang. *error*) rozumie się jako przejaw defektu w systemie komputerowym. Z kolei **defekt** (ang. *fault*), będący przyczyną błędu, jest jednocześnie konsekwencją awarii (uszkodzenia) innego systemu mającego wpływ na system rozważany. Wynika stąd, że awarie powstają jako skutki błędów, które są skutkami defektów, będących z kolei skutkami innych awarii itd. Powstaje w ten sposób łańcuch przyczynowo-skutkowy awarii, defektów, błędów, nowych awarii itd.

Zbliżone podejście do wymienionych pojęć usankcjonowano już w niedawno zatwierdzonej normie terminologicznej organizacji IEEE [2], podając następujące definicje:
awaria (ang. *failure*) – *niezdolność systemu lub jego składowej do wykonywania wymaganych funkcji pod względem wydajności,*
defekt (ang. *fault*) – *wada sprzętu lub niepoprawny krok, proces lub definicja danych w programie komputerowym,*
błąd (ang. *error*) – *różnica między obliczoną, zaobserwowaną lub zmierzoną wartością lub warunkiem a prawdziwą, założoną lub teoretycznie poprawną wartością lub warunkiem,*
omyłka (ang. *mistake*) – *działanie ludzkie prowadzące do niepoprawnego wyniku.*
W użyciu potocznym, wszystkie wymienione pojęcia bywają nazywane błędami, często tak też jest – choć nie powinno – w technice. Należy rozróżniać między działaniem ludzkim (**omyłką**), jej przejawem w sprzęcie lub oprogramowaniu (**defektem**), skutkiem tego defektu (**awarią**) i otrzymanym wskutek awarii niepoprawnym wynikiem (**błędem**). Ważnym i często występującym rodzajem awarii jest tzw. załamanie systemu komputerowego zdefiniowane następująco [2]:
załamanie (ang. *crash*) – nagła i całkowita awaria systemu komputerowego lub jego składowej.

Konsekwencje różnego rozumienia pojęcia błędu znajdują też odzwierciedlenie w nazwach i definicjach odpowiednich cech systemu komputerowego, związanych z jego rzetelnością, np. [2]:
tolerowanie błędów (ang. *error tolerance*) – zdolność systemu lub jego składowych do kontynuacji normalnego działania mimo obecności błędnych danych wejściowych,
tolerowanie defektów (ang. *fault tolerance*) – *zdolność systemu lub jego składowych do kontynuacji normalnego działania mimo obecności defektów sprzętowych lub programowych.*

Ściśle związane z wymienionymi właściwościami są też inne aspekty rzetelności:
odporność (ang. *robustness*) – *stopień, w jakim system lub jego składowe mogą funkcjonować poprawnie w obecności nieważnych sygnałów wejściowych lub uciążliwych warunków środowiskowych* (definicja IEEE Std 610.12);
integralność (ang. *integrity*) – *stopień, w jakim system lub jego składowe zapobiegają nieuprawnionemu użyciu lub zmodyfikowaniu programów komputerowych lub danych* (IEEE Std 610.12);
dyspozycyjność (ang. *availability*) – *zdolność jednostki funkcjonalnej do wykonywania wymaganej funkcji w określonych warunkach, w określonej chwili lub przedziale czasu, przy założeniu, że dostępne są wymagane zasoby zewnętrzne* (IEV 191 [3];
nadmiarowość (ang. *redundancy*) – *zapewnienie alternatywnych (identycznych lub różnych) elementów lub systemów tak, aby jeden mógł wykonywać pożądaną funkcję niezależnie od sprawności lub awarii innych* (EWICS TC7);
zróżnicowanie (ang. *diversity*) – *istnienie różnych sposobów wykonywania wymaganej funkcji* (IEV 191, EWICS TC7).

Na zakończenie warto dodać, że tę tematykę poruszano już na łamach Informatyki, w numerach 11 i 12 z 1983 roku.

JANUSZ ZALEWSKI

LITERATURA

[1] Avizienis A., Laprie J. C.: Dependable Computing – From Concepts to Design Diversity. Proc. of the IEEE, Vol. 74, No. 5, pp. 629–638, 1986
[2] Institute of Electrical and Electronics Engineers: Glossary of Software Engineering Terminology. IEEE Std 610.12, New York, 1989
[3] International Electrotechnical Commission: International Electrotechnical Vocabulary. Reliability, Maintainability and Quality of Service. IEV 191, Geneva, 1986.

Ogłoszenia ● Ogłoszenia ● Ogłoszenia ● Ogłoszenia

Spółdzielnia Rzemieślnicza „ELMECH", ul. Dobra 56, 00–312 Warszawa, oferuje OWIJARKI ELEKTRYCZNE (pistoletowe) do połączeń „wire-wrap", przystosowane do druku ø 0,25–0,35 mm. Informacje – telefon 22-94-46.

EO/494/89

Ogłoszenia ● Ogłoszenia ● Ogłoszenia ● Ogłoszenia

SWEDEX UNIVERSAL Co.

PRZEDSIĘBIORSTWO ZAGRANICZNE W POLSCE

Biuro: ul. Kolejowa 2, 05–250 Radzymin
Teleks 815888 swedx pl, telefon: Warszawa 762004 w. 13 lub 365
Zakład Elektroniki: ul. Wspólna 1, 05–440 Wesoła-Zielona
Teleks 813935 swedx pl, telefon: Warszawa 153365

OFERTA ZAKŁADU ELEKTRONIKI

MODEM TRANSMISJI DANYCH 1230P

- umożliwia transmisję dwukierunkową jednoczesną (full duplex) po łączach telefonicznych komutowanych i dzierżawionych: w trybie asynchronicznym z szybkościami 0–300 b/s, 600 b/s i 1200 b/s; w trybie synchronicznym z szybkością 1200 b/s;
- zgodny z zaleceniami CCITT V.21, V.22;
- posiada świadectwo homologacji Instytutu Łączności PRL.

KONCENTRATOR STANOWISK OPERATORSKICH KSO-4 DO REJESTRATORA DANYCH MERA 9150 (SEECHEK)

- pozwala na dołączenie do Mery 9150 poprzez jedno łącze transmisji danych do 4 stanowisk operatorskich i drukarki systemowej;
- nie wymaga zmian w systemie operacyjnym i sprzęcie rejestratora;
- posiada wbudowane testy pozwalające na szybką lokalizację uszkodzeń i sprawdzenie łącza transmisji danych;
- jest produkowany także w wersji z wbudowanym modemem stałoprądowym KN-9600.

ADAPTER TELEGRAFICZNY ATG-2 DO MERY 9150

- umożliwia wprowadzanie danych i teleksów bezpośrednio z dalekopisu (sieci teleksowej) do Mery 9150 z kontrolą i automatycznym przesyłaniem komunikatów (teleksów) zwrotnych.

ADAPTER TELEKOMUNIKACYJNY AT 8-MPX (UPD asynchroniczne i synchroniczne)

- pozwala na dołączenie do EMC ODRA 1305 poprzez multiplekser MPX 325 ośmiu zdalnych terminali np. drukarki DZM 180 KSRE, monitora ekranowego ICL 7181 lub odpowiednika produkcji krajowej, stacji ICL 7020, mini- lub mikrokomputerów z emulatorami powyższych terminali;
- realizuje automatyczne rozłączenie połączeń komutowanych.

INTERFEJSY POMIAROWE SU-GPIB I SU-GPIBF STANOWIĄCE PEŁNĄ IMPLEMENTACJĘ NORMY IEC-625 (IEEE 488) UMOŻLIWIAJĄCE STEROWANIE PROCESEM POMIAROWYM Z SZYBKOŚCIĄ DO 500 kB/s

- oprogramowanie: procedury wywoływane z takich języków programowania, jak C, PASCAL, FORTRAN, BASIC;
- układ interfejsu zrealizowany jest w oparciu o kontroler 7210 oraz bufory firmy Texas Instruments.

PAKIET TRANSMISJI SZEREGOWEJ SYNCHRONICZNEJ/ASYNCHRONICZNEJ BSC DO KOMPUTERÓW KOMPATYBILNYCH Z IBM PC XT/AT

- umożliwia pracę komputera jako terminala np. ICL 7181, ICL 7020, IBM 2780, IBM 3780, IBM 3270;
- dostarczamy oprogramowanie emulacyjne pakietu BSC.

PROGRAMATOR PAMIĘCI EPROM ORAZ MIKROKOMPUTERÓW JEDNOUKŁADOWYCH

- umożliwia programowanie pamięci typu 2716--27512 i ich funkcjonalnych odpowiedników o napięciach programujących 12,5; 21; 25 V oraz mikrokomputerów jednoukładowych 8741, 8748, 8749, 8751, 8755;
- automatycznie wybiera parametry programowania;
- współpracuje z komputerem przez styk RS-232 (V.24);
- oprogramowanie do komputerów IBM PC XT/AT.

KOMPUTERY SU 88-PC W PEŁNI KOMPATYBILNE Z IBM PC KONFIGUROWANE ZGODNIE Z ŻYCZENIEM KLIENTA.

EO/1263/88

Redmill F.: Najnowsze zalecenia dotyczące bezpieczeństwa i niezawodności systemów komputerowych

INFORMATYKA 1989, nr 7, s. 1

Charakterystyka działalności Komitetu Technicznego nr 7 międzynarodowej organizacji EWICS oraz jego zaleceń dotyczących bezpieczeństwa i niezawodności eksploatacji systemów komputerowych.

Redmill F.: The latest recomendations for safety and reliability of computer systems

INFORMATYKA 1989, No. 7, p. 1

Characteristics of the EWICS Technical Committee No. 7 activity and its recomendations for safety and reliability of computer system operation.

Redmill F.: Die neuesten Empfelungen für Sicherheit und Zuverlässigkeit der Computersysteme

INFORMATYKA 1989, Nr. 7, S. 1

Eine Charakteristik von Tätigkeit des Technischen Ausschusses Nr. 7 in internationaler Organisation EWICS und seiner Empfelungen für Sicherheit und Betriebszuverlässigkeit der Computersysteme.

Zalewski J.: Rzetelność systemów komputerowych w zastosowaniach krytycznych

INFORMATYKA 1989, nr 7, s. 4

Charakterystyka problemu oraz analiza książki na temat niezawodności działania systemów komputerowych w zastosowaniach krytycznych.

Zalewski J.: Dependability of computer systems in critical applications

INFORMATYKA 1989, No. 7, p. 4

Characteristics of the problem and analysis of the book on realiability of computer systems operation in critical applications.

Zalewski J.: Echtheit der Computersysteme in kritischen Anwendungen

INFORMATYKA 1989, Nr. 7, S. 4

Eine Charakteristik des Problems und Analyse des Buches über Betriebszuverlässigkeit der Computersysteme in kritischen Anwendungen.

Gondzio M., Pawłowski M., Wytrębowicz J.: Uruchamianie urządzeń mikroprogramowanych (1)

INFORMATYKA 1989, nr 7, s. 6

Pierwsza część charakterystyki problemu wspomagania procesu uruchamiania urządzeń mikroprogramowanych, zawierająca omówienie narzędzi opracowanych i stosowanych w Instytucie Informatyki Politechniki Warszawskiej.

Gondzio M., Pawłowski M., Wytrębowicz J.: Developing of microprogrammed devices (1)

INFORMATYKA 1989, No. 7, p. 6

First part of characteristics of the microprogrammed devices assisted development process, which includes discussion of the in the Informatics Institute of Warsaw Technical University elaborated and applied tools.

Gondzio M., Pawłowski M., Wytrębowicz J.: Inbetriebsetzung der mikroprogrammierten Einrichtungen (1)

INFORMATYKA 1989, Nr. 7, S. 6

Erster Teil einer Charakteristik der Unterstützung von Inbetriebsetzungsprozess der mikroprogrammierten Einrichtungen, der eine Besprechung der im Institut für Informatik der Warschauen Technischen Universität erarbeiteten und angewendeten Werkzeuge umfasst.

Kuraś J., Lembas J., Skomorowski M.: Język symulacji systemów o działaniu ciągłym ze zdarzeniami dyskretnymi

INFORMATYKA 1989, nr 7, s. 9

Szczegółowa charakterystyka rozwiązań języka GODYS-PC, przeznaczonego do symulacji systemów o działaniu ciągłym ze zdarzeniami dyskretnymi na mikrokomputerach klasy IBM PC.

Kuraś J., Lembas J., Skomorowski M.: Simulation language for systems with continuous operation and discrete events

INFORMATYKA 1989, No. 7, p. 9

Detailed characteristics of the GODYS-PC language solutions, which is appropriated for simulation of systems with continuous operation and discrete events on microcomputers of the IBM PC category.

Kuraś J., Lembas J., Skomorowski M.: Simulierungssprache für Systeme mit fortlaufender Wirkung und diskreten Ereignissen

INFORMATYKA 1989, Nr. 7, S. 9

Eine detaillierte Charakteristik von Lösungen der GODYS-PC-Sprache, die für Simulierung der Systeme mit fortlaufender Wirkung und diskreten Ereignissen auf Mikrorechner der IBM PC-Klase bestimmt ist.

Tataj M., Toruń A.: Tomografia komputerowa

INFORMATYKA 1989, nr 7, s. 12

Charakterystyka rozwoju oraz sposobu działania tomografii komputerowej w zastosowaniach medycznych.

Tataj M., Toruń A.: Computer tomography

INFORMATYKA 1989, No. 7, p. 12

Characteristics of computer tomography development and its operation method in medical applications.

Tataj M., Toruń A.: Computertomographie

INFORMATYKA 1989, Nr. 7, S. 12

Eine Charakteristik von Eintwicklung und Wirkungsmethoden der Computertomographie in medizinischen Anwendungen.

Iszkowski W.: Przyspieszanie obliczeń przez ich rozpraszanie (3). Podsumowanie

INFORMATYKA 1989, nr 7, s. 15

Dokończenie przeglądu obecnie stosowanych metod zwiększania szybkości działania komputerów, zawierające podsumowanie rozważań oraz wnioski autora.

Iszkowski W.: Acceleration of computations through scattering (3). Recapitulation

INFORMATYKA 1989, No. 7, p. 15

Termination of survey of the to-day used methods for computer speed increasing, which includes recapitulation of considerations and conclusions of the author.

Iszkowski W.: Beschleunigung von Berechnungen durch Zerstreung (3). Zusammenfassung

INFORMATYKA 1989, Nr. 7, S. 15

Beendigung eines Übersichts von heutzutage angewendeten Methoden für Computergeschwindigkeitssteigerung, der eine Zusammenfassung von Erwägungen und die Schlussfolgegungen des Autors umfasst.

Szczerbiński Z.: Nowa generacja superkomputerów Cray (2). Cray-2

INFORMATYKA 1989, nr 7, s. 19

Druga część charakterystyki nowej generacji superkomputerów Cray, zawierająca omówienie architektury, technologii i oprogramowania modelu Cray-2, a także perspektyw dalszego rozwoju tej rodziny komputerów.

Szczerbiński Z.: New generation of the Cray supercomputer (2). The Cray-2

INFORMATYKA 1989, No. 7, p. 19

Second part of characteristics of the Cray supercomputer new generation, which includes discussion of the Cray - -2 model architecture, technology and software, as well as of this computer family further development.

Szczerbiński Z.: Neue Generation von Cray-Supercomputern (2). Der Cray-2

INFORMATYKA 1989, Nr. 7, S. 19

Zweiter Teil einer Charakteristik von neuen Generation des Cray-Supercomputers, der eine Besprechung von Architektur, Technologie und Software des Cray-2-Modells, sowie von Entwicklungsaussichten dieser Computerfamilie, umfasst.

WYDAWNICTWA NAUKOWO-TECHNICZNE

Roman Świnarski: System operacyjny CP/M. Warszawa 1980, wydanie 1, nakład 20 tys., s. 200, cena 650 zł, ISBN 83-204-1022-3

W książce opisano system operacyjny CP/M-80 (wersja 2.2) – jeden z najbardziej rozpowszechnionych systemów operacyjnych dla mikrokomputerów 8-bitowych. Zawiera ona szczegółowy opis samego systemu, jego struktury, dyrektyw, zasad organizacji danych, a także praktyczne przykłady tworzenia, uruchamiania i wykonywania pod kontrolą CP/M programów w Basicu, Fortranie, Pascalu i języku asemblera.
Książka jest przeznaczona dla szerokiego kręgu Czytelników zainteresowanych programowaniem komputerów 8-bitowych.

Roman A. Plaza, Eugeniusz J. Wróbel: Systemy czasu rzeczywistego. Warszawa 1988, wydanie 1, nakład 7 tys. egz., s. 190, cena 600 zł, ISBN 83-204-1064-9

W książce przedstawiono zagadnienie związane z konstrukcją, oprogramowaniem i uruchamianiem mikrokomputerowych systemów czasu rzeczywistego. Specyfikę systemów czasu rzeczywistego przedstawiono na przykładzie uniwersalnego, wielomodułowego mikrokomputerowego systemu przeznaczonego do sterowania procesów przemysłowych. Szczególną uwagę zwrócono na układy łączące system z obiektem przemysłowym. Podano przykłady konkretnych rozwiązań systemów przemysłowych.
Książka jest przeznaczona dla inżynierów i techników projektujących systemy oraz dla studentów wyższych szkół technicznych.

Donald Hearn, M. Pauline Baker: Grafika mikrokomputerowa – metody i zastosowania (tłum. M. Łakomy, J. Zabrodzki). Warszawa 1988, wydanie 1, nakład 30 tys. egz., s. 310, cena 1100 zł, ISBN 83-204-0988-8

W książce zawarto zbiór podstawowych wiadomości z grafiki komputerowej. Zaprezentowano różne metody rysowania, transformacji i animacji obrazów. Omówiono istotę grafiki trójwymiarowej. Prezentację różnorodnych zastosowań grafiki w nauczaniu, domu, handlu itp. wzbogacono licznymi przykładami programów. Praca zawiera wiele rozwiązań praktycznych.
Jest przeznaczona dla szerokiego kręgu odbiorców.

Wojciech Cellary, Zbyszko Królikowski: Wprowadzenie do projektowania baz danych dBase III. Warszawa 1988, wydanie 1, nakład 30 tys. egz., s. 180, cena 540 zł, ISBN 83-204-1089-4

Książka zawiera wprowadzenie do teorii relacyjnych baz danych oraz omówienie zasad posługiwania się popularnym systemem dBase III. W części teoretycznej przedstawiono trzy podstawowe operacje relacyjne – projekcję, selekcję i połączenie – oraz podstawy projektowania relacyjnych baz danych, ze szczególnym uwzględnieniem tzw. procesu normalizacji. Część praktyczna jest poświęcona omówieniu systemu dBase III i jego poleceń: zakładania, modyfikowania, aktualizowania i porządkowania baz danych. Liczne przykłady ułatwiają zrozumienie wprowadzonych pojęć oraz działanie poszczególnych poleceń.
Książka jest przeznaczona dla wszystkich zainteresowanych używaniem systemu dBase III, zwłaszcza dla projektantów systemów baz danych realizujących konkretne zastosowania.

Ryszard K. Kott: Programowanie w języku Pascal. Warszawa 1988, wydanie 1, nakład 30 tys. egz., s. 256, cena 780 zł, ISBN 83-204,0946-2

Książka zawiera przystępny kurs programowania w Pascalu. Wszystkie pojęcia omówiono od podstaw, wprowadzając w pierwszej części książki tylko niektóre elementy języka. Druga natomiast część zawiera pełny opis wszystkich, także zaawansowanych, konstrukcji Pascala (zgodnie z normą międzynarodową ISO nr 7185). Materiał zilustrowano licznymi przykładami, zamieszczono także zestaw pytań i zadań dla każdego rozdziału. Dodatki zawierają m.in. odpowiedzi i pełną składnię Pascala według oryginału normy.
Książka jest przeznaczona dla osób zainteresowanych programowaniem komputerów w praktyce. Może być także podręcznikiem programowania dla studentów i uczniów szkół średnich.

Jan Bielecki: Język Forth, Warszawa 1988, wydanie 1, nakład 9 tys. egz., s. 164, cena 580 zł, ISBN 83-204-0930-6

W książce opisano język programowania Forth. Podano pojęcia podstawowe dotyczące tego języka, przedstawiono operacje stosowe i słownikowe oraz omówiono zasady definiowania operatorów, kompilatorów i instrukcji. Przedstawiono możliwości rozbudowania języka, współpracę z urządzeniami zewnętrznymi i oprogramowanie grafiki.
Książka jest przeznaczona dla Czytelników interesujących się programowaniem mikrokomputerów.

Keith L. Clark, Frank G., McCabe: Mikro-Prolog (tłum. P. Luboński). Warszawa 1988, wydanie 1, nakład 13 tys. egz., s. 356, cena 700 zł, ISBN 83-204-0972-1

Przedmiotem książki jest język programowania micro-Prolog, stanowiący wersję Prologu przeznaczoną dla mikrokomputerów. Prolog jest językiem programowania nowej generacji, opartym na logice formalnej. Micro-Prolog różni się od wcześniejszych wersji Prologu składnią zbliżoną do naturalnego języka angielskiego. Książka zawiera obszerne wprowadzenie do programowania w micro-Prologu, ze szczególnym naciskiem na zastosowania z dziedziny baz danych, przetwarzanie list oraz przetwarzanie języka naturalnego. Omówiono także sposób wykonywania programu przez interpretator i związane z tym zagadnienia efektywności oraz zaawansowane techniki programowania pozwalające m.in. pisać samomodyfikujące się programy.
Książka nie wymaga od czytelnika wiedzy z informatyki, może więc służyć nie tylko zawodowym informatykom i studentom kierunków informatycznych, lecz także wszystkim zainteresowanym programowaniem komputerów.

PRZEDSIĘBIORSTWO
ZASTOSOWAŃ INFORMATYKI
meditronik

oferuje:
- systemy mikrokomputerowe
- programy aplikacyjne dla różnych dziedzin gospodarki (na życzenie wysyłamy katalog)
- poszukiwane komponenty elektroniczne
- interfejs do kamery video (opc. CCD) z bogatą biblioteką oprogramowania
- emulator Z80
- tester układów scalonych i pamięci
- programator BPROM
- asynchroniczny procesor komunikacyjny
- konwerter RS-232 – Centronics

instaluje:
- połączenia międzykomputerowe (XT/AT – ODRA/RIAD/IBM)
- systemy sieciowe (NOVELL)
- systemy wielodostępne (SCO Xenix 286, 386, Unix System V)

Jeżeli jesteś autorem oryginalnego programu aplikacyjnego
- skontaktuj się z nami, będziemy pośredniczyć w sprzedaży Twojego programu, dbając o ochronę Twoich praw autorskich!

Nasz adres:

00–194 Warszawa, ul. Dzika 4
tel. (02) 635-22-63, 635-22-64
fax (02) 635-21-95
tlx 816075 medi pl

EO/605/89

Tajwan na rynku informatycznym

Tajwan zajmuje tylko 36 tys. km² i liczy 18 milionów ludności, jednakże już w 1986 roku wysunął się na siódme miejsce wśród największych producentów sprzętu informatycznego w świecie, wykazując najwyższe (64%) tempo wzrostu wśród pierwszej dziesiątki tych producentów. Jak podano w tabeli 1 wartość produkcji sprzętu informatycznego Tajwanu wzrosła z 1,3 mld dolarów w 1985 r. do 2,1 mld w 1986 r., a na rok 1987 przewidywano 3 mld dol., co odpowiada 43% wzrostu w stosunku do roku poprzedniego. Z krajów poza pierwszą dziesiątką tylko Korea Południowa osiągnęła w 1986 r. 48,4% wzrostu. Produkcja Tajwanu w zakresie sprzętu komputerowego stanowiła w 1986 r. 1,45% produkcji światowej.

Wśród poszczególnych działów sprzętu w tym samym roku wartość produkcji (w dolarach) stanowiła:

- mikroprocesory 455 mln,
- pakiety 800 mln,
- monitory 500 mln,
- terminale 318 mln,
- stacje dysków elastycznych 72,2 mln,
- drukarki 41,4 mln,
- inne urządzenia peryferyjne 45,5 mln.

Wartość eksportu w tych dziedzinach przedstawiono w tabeli 2.

Produkcja jest nastawiona w znacznym stopniu na eksport. Wywozi się za granicę 86,2% mikrokomputerów i ponad 98% monitorów, terminali, drukarek, czytników dyskietek, pakietów itd.
W tabeli 3 przedstawiono udział produkcji i eksportu tajwańskiego w rynku światowym.

Eksport sprzętu informatycznego w 1986 r., wynoszący 2 mld dolarów, stanowił 1,37% całego eksportu Tajwanu, a przewidywania na rok 1987 mówiły o wzroście tego wskaźnika do 1,75%. Większość nabywców informatycznego sprzętu tajwańskiego stanowili konstruktorzy zagraniczni, którzy zainwestowali w przedsiębiorstwa tajwańskie (44% w 1986 r.) i firmy kompletujące systemy (39%). Ta ostatnia grupa obecnie nieco zwiększa swój udział. Połowa tego eksportu idzie do USA, a Francja jest na ósmym miejscu (za Holandią, RFN, W. Brytanią, Kanadą, Austrią i Hong Kongiem), ale w latach 1985-1986 wykazała ona największy wzrost z 12,2 do 41,6 mln dolarów.

Przewidywania na 1991 rok mówią o wzroście produkcji sprzętu informatycznego Tajwanu do wartości rzędu 5 mld dolarów, co powinno dać mu szóste miejsce na świecie tuż za Francją.

Tabela 1. Wartość produkcji dziesięciu głównych producentów sprzętu informatycznego

Kraj	Wartość produkcji 1985 r. (mld dol.)	Pozycja 1985 r.	Wartość produkcji 1986 r. (mld dol.)	Pozycja 1986 r.	Procent wzrostu
USA	48,6	1	49,1	1	1,0
Japonia	17,1	2	19,3	2	12,8
RFN	5,3	3	5,8	3	9,4
W. Brytania	4,2	4	4,6	4	9,5
Francja	3,4	5	3,8	5	11,7
Włochy	2,9	6	3,1	6	7,0
Tajwan	1,3	9	2,1	7	64,0
Irlandia	1,5	7	1,7	8	13,3
Holandia	1,5	7	1,6	9	6,6
Singapur	1,22	10	1,36	10	11,5

Tabela 2. Eksport sprzętu z Tajwanu w latach 1983–1987

Rodzaj sprzętu	1983	1984	1985	1986		1987 (prognoza)		
	Wartość (mln dol.)	Wartość (mln dol.)	Wartość (mln dol.)	Liczba (tys.)	Wartość (mln dol.)	Liczba (tys.)	Wartość (mln dol.)	Wzrost (%)
Mikrokomputery	12	152,2	240	1113,4	393	1600	570	45
Napęd dyskietek	15,2	85,8	42	716,6	71	1000	100	40
Drukarki	12	22,9	45	83,9	41	80	40	-5
Terminale	96,7	207,1	225	1318,3	317	1600	400	25
Monitory	130,8	319	303	3972,7	500	50000	700	40
Inne urządzenia peryferyjne	22,4	104,4	256		44		70	45
Łącznie	**418,1**	**1004,8**	**1220**		**2063**		**2900**	

Tabela 3. Rynek światowy a produkcja i eksport Tajwanu

		1983	1984	1985	1986
Rynek światowy	Wartość (mld dol.)	97,0	112,0	127,0	142,0
	Procent wzrostu	13,7	16,0	13,0	11,9
Produkcja Tajwanu	Wartość (mld dol.)			1,3	2,13
	Procent wzrostu			21,0	64,0
	Udział w rynku światowym (%)	0,43	0,89	0,96	1,45
Eksport Tajwanu	Wartość (mld dol.)	0,418	1,0	1,22	2,0
	Procent wzrostu	161,0	140,0	21,0	69,0

Sieci MAP

Dotychczas poszczególne oddziały firmy Hewlett-Packard stosowały własne sieci w swych systemach. Obecnie firma zgłosiła akces do grupy użytkowników protokołu MAP (*Manufacturing Automation Protocol* – protokół automatyzacji wytwarzania) modelu OSI, ogłaszając szereg wyrobów, które pojawiają się na rynku w najbliższym okresie. Pierwszą kartą ma być MAP 3.0 MMS (*Manufacturing Message Specification* – specyfikacja komunikatów wytwarzania). Oprogramowanie i zespół narzędzi dotyczących systemu sprzężenia z urządzeniem opracowano w końcu 1988 roku, a analizator protokołów MAP 3.0 i oprogramowanie do przesyłania, dostępu i zarządzania plikami – w połowie 1989 roku.

Kostki RAM

Firma Toshiba America, z Irvine w Kalifornii, zaoferowała na wystawie Electro'88 kostkę pamięci operacyjnej o konfiguracji 128 K × 8 bitów, poborze mocy 350 mW i czasie dostępu 70-100 ns (zależnie od wersji). Podobne oferty z innych firm są oczekiwane w najbliższym czasie.

Układy logiczne

Nowa rodzina układów logicznych firmy Sony osiąga częstotliwość przełączania 4 GHz przy utrzymaniu poziomu wydzielanej mocy poniżej 1 W. Są one realizowane w technice ECLIII, gdzie odległość ścieżek jest rzędu 1 μm, a częstotliwość przejścia 10 GHz. Opóźnienie na bramkę przy prądzie 0,3 mA wynosi 80 ps, tj. mniej niż dla arsenku galu. Opracowywana jest technika ECLIV, która przy odległościach ścieżek poniżej mikrona powinna być dwukrotnie szybsza.

Oprac. JAN RYŻKO